Publication Series of the John von Neumann Institute for Computing (NIC)
NIC Series Volume 16

John von Neumann Institute for Computing (NIC)

Volker Sander

Design and Evaluation of a Bandwidth Broker that Provides Network Quality of Service for Grid Applications

NIC Series Volume 16 ISBN 3-00-010002-4

Central Institute for Applied Mathematics

Die Deutsche Bibliothek – CIP-Cataloguing-in-Publication-Data
A catalogue record for this publication is available from Die Deutsche Bibliothek

Publisher: NIC-Directors
Distributor: NIC-Secretariat
Research Centre Jülich
52425 Jülich
Germany
Internet: www.fz-juelich.de/nic
Printer: Graphische Betriebe, Forschungszentrum Jülich

NIC Series Volume 16 ISBN 3-00-010002-4

Design and Evaluation of a Bandwidth Broker that Provides Network Quality of Service for Grid Applications

Von der Fakultät für Elektrotechnik und Informationstechnik der
Rheinisch-Westfälischen Technischen Hochschule Aachen
zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften
genehmigte Dissertation

vorgelegt von

Diplom-Informatiker

Volker Sander

aus Bottrop

Berichter: Universitätsprofessor Dr. rer. nat. F. Hoßfeld
Universitätsprofessor Dr. rer. nat. O. Spaniol

Tag der mündlichen Prüfung: 27.11.2002

Abstract

The term computational Grid is used to describe a problem solving environment in which geographically and organizationally dispersed resources are integrated into a common infrastructure. Building computational Grids requires the existence of a resource management framework that provides a unique and secure interface to the reservation and allocation capabilities offered by the various local resource management systems. In this context, the properties of the interconnecting network are of major importance, because the actual service parameters such as bandwidth or latency are a result of the cooperation of all related network domains.

The provision of service guarantees over an IP-based network can be achieved by applying a different forwarding treatment for selected packets. The Differentiated Services architecture defines a framework for implementing a scalable service differentiation in the existing Internet by an aggregation of flows to a small number of different traffic classes with a particular forwarding treatment. While this concept allows an efficient provision of service guarantees, the actual service parameters depend on the dynamic composition of the traffic classes. Here, a specific management entity, called bandwidth broker, comes into place. A bandwidth broker controls the dynamic access to traffic classes for a single trust domain and can thus be viewed as resource management system for the resource "network". The integration of a bandwidth broker into a general resource management framework of computational Grids is therefore a natural extension.

This dissertation evolves a flexible bandwidth broker architecture with the goal to incorporate the network as a manageable resource into a Grid resource management infrastructure. The presented framework intermediates between the unique requirements of emerging Grid applications and the applicable forwarding treatment of packets by considering the complex trust relationships and usage policies that can apply in a multi-domain network environment. The accomplishment of the approach is the secure and transparent end-to-end integration of a composition of bilaterally interrelated bandwidth brokers. The implementation of a prototype and a thorough experimental evaluation validates the feasibility and flexibility of the architectural design.

Kurzfassung

Computational Grids verknüpfen die Ressourcen geographisch verteilter, meist administrativ unabhängiger Einrichtungen und integrieren diese zu einer konsistenten Infrastruktur. Zur effizienten Umsetzung dieser Aufgabe ist ein Ressourcen-Verwaltungssystem erforderlich, welches die lokal angebotenen Reservierungs- und Zuweisungsmechanismen nutzt und die Dienste über eine einheitliche und sichere Schnittstelle im Computational Grid zur Verfügung stellt. Der Ressource “Netzwerk” kommt dabei eine besondere Bedeutung zu, da sich Dienstgarantien wie für Bandbreite oder Latenz aus dem planbaren Zusammenwirken der Eigenschaften aller betroffenen Netzwerkdomänen bestimmen.

Dienstgarantien in IP-basierten Netzwerken können u.a. durch eine differenzierte Behandlung der Datenpakete in den einzelnen Routern erreicht werden. Um den Differenzierungsaufwand auch in komplexen Netzwerken zu begrenzen, klassifiziert die Differentiated-Services-Architektur Pakete in sogenannte Verkehrsaggregate und spezifiziert Regeln zu deren Behandlung. Mit diesem Verfahren erzielbare Dienstgarantien hängen nicht nur von den zugewiesenen Behandlungsregeln, sondern auch von der sich laufend verändernden Struktur und Zusammensetzung der Verkehrsaggregate ab. Der damit verbundene dynamische Zugang zu den Verkehrsaggregaten wird durch eine spezielle Softwareschicht - den Bandwidth Broker - verwirklicht. Diese Softwareschicht reguliert die effektive Nutzung der Dienstgarantien einer Netzwerkdomäne und sollte daher in die einheitliche Schnittstelle zur Ressourcenanforderung von Computational Grids integriert werden.

Die Dissertationsschrift spezifiziert eine Bandwidth-Broker-Architektur mit der Absicht, die Ressource “Netzwerk” als planbare Größe in die Infrastruktur des Computational Grid einzubetten. Ausgehend von den typischen Anwendungsklassen werden die spezifischen Dienstanforderungen identifiziert und in der Architektur berücksichtigt. Der vorgestellte Entwurf stellt eine vermittelnde Schicht zwischen Anforderungen und Behandlungsregeln dar und berücksichtigt bei dieser Aufgabe die komplexen Rahmenbedingungen eines in administrativ unabhängige Domänen strukturierten Netzwerks. Damit wird eine transparente und sichere Ende-zu-Ende-Integration eines Verbundes aus bilateral in Beziehung stehender Bandwidth Brokers erreicht. Die Implementierung eines Prototyps zeigt die technische Realisierbarkeit des vorgestellten Entwurfs und bestätigt in Experimenten dessen Flexibilität auch für heterogene Verkehrsaggregate.

Acknowledgements

This thesis was written at the Zentralinstitut für Angewandte Mathematik (ZAM) of Forschungszentrum Jülich. I would like to express my gratitude to the many people who have helped me directly or indirectly with this dissertation.

First and foremost, I would like to thank my advisor, Prof. Dr. Friedel Hoßfeld, who has the chair of Technical Informatics and Computer Science at Aachen University of Technology and who was director of the ZAM until end of July 2002. This dissertation would not have been possible without his invaluable advice and continuous support. I also want to express my gratitude to Prof. Dr. Otto Spaniol for his constructive criticisms and for serving as the second referee. In addition, I would like to acknowledge Dietmar Erwin, head of the ZAM department Operating Systems, for his support and suggestions.

I am also deeply indebted to Prof. Dr. Ian Foster, University of Chicago and Argonne National Laboratory, for his effort in guiding and supporting my studies at Argonne National Laboratory. His suggestions, ideas and comments were more than helpful to me and did never stop.

Special thanks to my friends Dr. Alain Roy and Markus Fidler for their endless time, inspiration and incredible support and help. It was and is an honor to collaborate with them.

I would also like to express my appreciation to Linda Winkler and William A. Adamson for their suggestions and contributions, to Philip Wieder and Achim Streit for reading and commenting on drafts of this dissertation, to Sabine Werner and Olaf Mextorf for their assistance, and to Steven Tuecke and Karl Czajkowski for all the technical discussions we had. Furthermore, thanks to all the other friends, colleagues and collaborators who assisted and encouraged me during my studies.

Above all, I would like to express my deepest gratitude to the people I love and to whom I would like to dedicate this dissertation: My lovely wife Petra and my two little children Lea and Lara.

Volker Sander
December 2002, Jülich, Germany

Contents

1 Introduction **1**

2 Computational Grids **5**

2.1 Future Problem Solving Infrastructures 5

2.2 Building Blocks for Computational Grids 9

2.2.1 Core Services 10

2.2.2 High-Level Services 12

2.3 Grid Applications 13

2.3.1 Distributed Supercomputing 13

2.3.2 High-throughput Computing 16

2.3.3 On-demand Computing 17

2.3.4 Data-intensive Computing 18

2.3.5 Collaborative Computing 19

2.4 Conclusion 21

3 IP-based Quality of Service **23**

3.1 Overview 23

3.2 IP-based QoS Architectures 25

3.2.1 The Integrated Services Framework 25

3.2.2 The Differentiated Services Architecture 29

3.3 Multiprotocol Label Switching 35

3.4 Network Calculus: Formal Aspects of Network QoS 40

3.5 Conclusion 45

4 Requirements for a Grid Bandwidth Broker **47**
4.1 Integration into Computational Grids 47
4.1.1 Grid Resource Management 48
4.1.2 Coordinated Reservation and Allocation of Resources 48
4.1.3 Application Interface 49
4.1.4 Policy Information for Distributed Authorization 49
4.1.5 Deployability 51
4.2 Traffic Engineering 51
4.3 Advanced Networking Demands 53
4.3.1 The Transmission Control Protocol 53
4.3.2 The Berkeley Socket Interface 57
4.3.3 Bulk Transfers with Deadline Support 60
4.3.4 Control and Adaptation 61
4.4 Conclusion 61

5 Design of a Grid Bandwidth Broker **63**
5.1 Design Layout 63
5.1.1 Scope of Control 63
5.1.2 Basic Architectural Framework 65
5.1.3 External Signaling Interface 66
5.1.4 State Repository 68
5.1.5 Policy Repository 70
5.1.6 Control Procedures 71
5.1.7 Internal Signaling Interface 71
5.2 Support of a Coordinated Reservation and Allocation 72
5.3 Building Per-Domain Behaviors 75
5.4 Grid Resource Management Integration 90
5.4.1 User-Interface 90
5.4.2 Inter-domain Signaling and Authorization 91
5.5 Handling of Grid Applications 106
5.5.1 Distributed Supercomputing 106
5.5.2 High-throughput Computing 109

5.5.3 On-demand Computing 110
5.5.4 Data-intensive Computing 110
5.5.5 Collaborative Computing 111
5.6 Support of Adaptive Applications 112
5.6.1 Refined Architectural Framework 112
5.6.2 Bulk Transfer Support 114
5.7 Implementation Framework 114
5.8 Conclusion 118

6 Design Evaluation **121**
6.1 GARA: A Prototype for a Grid Bandwidth Broker 121
6.1.1 Overview 121
6.1.2 General Implementation 124
6.1.3 Bandwidth Broker Implementation 125
6.2 Experimental Setup 130
6.2.1 Testbeds 130
6.2.2 Evaluation Tools 132
6.3 Multi-Aggregate Environment 133
6.3.1 Implementation and Evaluation of the Best-Effort Service 133
6.3.2 Implementation and Evaluation of an Olympic Service 138
6.3.3 Implementation and Evaluation of a Premium Service 143
6.3.4 Proposed Assignment of Service Classes 144
6.4 Single-Aggregate Environment 145
6.4.1 Evaluation of Traffic Policing 145
6.4.2 Evaluation of Heterogeneous Aggregates 147
6.5 Applying MPLS 151
6.5.1 The Overhead of the On-demand Allocation of Tunnels 151
6.5.2 Link Protection 153
6.6 Evaluation of Advanced Services 156
6.6.1 Support of Bulk Transfers 156
6.6.2 Evaluation of TCP Pacing for a Guaranteed Rate Service 161
6.6.3 TCP Pacing for Distributed Supercomputer Applications 164
6.7 Conclusion 167

7 Conclusion **169**

List of Acronyms **171**

References **175**

List of Figures

2.1 Two different Grids built by two virtual organizations 6
2.2 A layered architectural model for computational Grids 10
2.3 Inherent burstiness of distributed supercomputing applications 15
2.4 Short-term impact of packet loss for bursty TCP applications 16
2.5 Traffic profile of a video-conferencing application 19

3.1 Integrated Services implementation reference model for routers 26
3.2 The basic principle of using RSVP for network reservations 28
3.3 Conceptual overview about the Differentiated Services model 30
3.4 The Random Early Detection algorithm. 35
3.5 Conceptual overview about an MPLS domain 36
3.6 Tunnel establishment in an MPLS domain 39
3.7 Arrival curve for Integrated Services flows 42
3.8 A rate-latency service curve 43
3.9 Delay and buffer boundaries for a token bucket constrained flow 45

4.1 Layer between network and Grid resource management 48
4.2 Multifarious service policies 50
4.3 Service limitations caused by standard shortest-path routing 52
4.4 Improved service provisioning by applying traffic engineering 53
4.5 Illustration of TCP's flow control 54
4.6 Description of TCP-Reno's window size evolution 56
4.7 Functional processing of TCP's three-way handshake 58

5.1 The multi-domain reservation problem 64
5.2 Functional decomposition of a bandwidth broker 67

5.3 Graphical representation of a slot table 69
5.4 The bandwidth broker embedded into the Grid resource management . . . 73
5.5 Illustration of the particular problem of aggregate scheduling 82
5.6 Multi-domain reservations with source-domain-based signaling 92
5.7 Consistency problem of source-domain-based signaling 93
5.8 Multi-domain reservations with hop-by-hop-based signaling 94
5.9 Reservation policy heterogeneity 97
5.10 A delegation model with capability certificates 104
5.11 Refinement of the proposed bandwidth broker architecture 112
5.12 Multi-service support for a single EF aggregate 117

6.1 Overview about GARA's multi-layered architecture 122
6.2 Application Programming Interfaces to GARA 123
6.3 Illustration of the use of GARA's end-to-end API 124
6.4 Illustration of GARA's implementation of a bandwidth broker 126
6.5 The behavior of TCP with an underestimated reservation 129
6.6 The GARNET testbed 131
6.7 The testbed at Forschungszentrum Jülich GmbH 131
6.8 Data transmission profile of a TV-news sequence under congestion 133
6.9 Transmission profile of the competing UDP best-effort flow 134
6.10 TCP file transfer when upstream congestion was applied 135
6.11 Overall impact of upstream congestion for TCP flows 135
6.12 Impact of upstream congestion on the round-trip time of packets 136
6.13 Throughput variation caused by upstream congestion 136
6.14 Short-term impact of upstream bursts 137
6.15 Packet loss due to upstream bursts 137
6.16 Traffic profile of a news sequence using the Bronze service 139
6.17 Traffic profile of a news sequence using the Silver service 139
6.18 Traffic profile of a news sequence using the Gold service 140
6.19 Traffic profile of a news sequence using the Premium service 140
6.20 Round-trip time of packets of a TCP Bronze file transfer 141
6.21 Throughput of a TCP Bronze file transfer 141

6.22 Round-trip time of Bronze packets with an increased window size 142
6.23 Throughput of a Bronze file transfer with an increased window size . . . 142
6.24 Round-trip time of packets of a TCP EF file transfer 143
6.25 Throughput of a TCP EF file transfer 144
6.26 Short-term impact of token bucket policing on the TCP throughput 146
6.27 Throughput of a TCP stream with an underestimated reservation 147
6.28 Behavior of a heterogeneous EF aggregate without traffic shaping 149
6.29 Behavior of a heterogeneous EF aggregate with shaping 150
6.30 Behavior of a heterogeneous EF aggregate with flow-based shaping . . . 151
6.31 Logical view of the attempted placement of LSPs 152
6.32 Signaling overhead associated with the setup of LSPs 152
6.33 Service interruption due to IP-convergence for link failure situations . . . 153
6.34 Packet delay variation due to IP-convergence for link failure situations . . 154
6.35 Lost packets due to IP-convergence for link failure situations 154
6.36 Service interruption when MPLS link protection is applied 155
6.37 Packet delay variation when link protection is applied 155
6.38 Deadline file transfer in a GR environment 157
6.39 Deadline file transfer in a wide area environment 158
6.40 Hierarchical file transfer applying for unused capacity 160
6.41 Hierarchical file transfer under congestion 160
6.42 TCP pacing compared to a regular TCP flow without SACK 162
6.43 TCP pacing compared to a regular TCP flow with SACK 163
6.44 Dynamic pacing of a TCP flow 163
6.45 Traffic profile of an MPI application when traffic shaping was applied . . 164
6.46 Single burst of an MPI application without traffic shaping 165
6.47 Single burst of an MPI application with strict traffic shaping 166
6.48 Single burst of an MPI application with a loose traffic shaping 167

List of Tables

2.1 QoS requirements of teleimmersion environments 21

5.1 Resource Allocation Request (RAR) message 95
5.2 Resource Allocation Answer (RAA) message 96

6.1 Core configuration of the Olympic classes 138
6.2 Mapping of applications to services in multi-aggregate environments . . . 144

Chapter 1

Introduction

The combination of multiple computational resources to solve a single problem has been focus of scientific research for many years. The idea of building a metacomputer, a virtual computer which encapsulates the underlying computing resources and which can be used to compute a single request on multiple distributed computers in parallel, was born in the early 1990s [118]. Based on the demand for being able to use a metacomputer, communication libraries evolved for massively parallel computer systems were extended to use standard Internet protocols for external communication. The performance capabilities of these Internet-based libraries became crucial for the efficient use of a metacomputer.

While the term metacomputer focused on coupling distributed computer resources to solve a single problem, in the late 1990s the term computational Grid—or "the Grid", as analogous to the electrical power Grid—was introduced as a generalization of the idea of a metacomputer to more general types of resources. Instead of focusing exclusively on computing resources, computational Grids are aimed to integrate all services required to solve scientific problems into one unique persistent infrastructure. This infrastructure is built by resources such as sensors, computational or virtual reality devices, data archives, as well as the interconnecting network.

Building computational Grids requires the existence of a sophisticated set of middleware. One major task of this software layer is to coordinate the simultaneous and/or ordered access to the underlying services built on top of potentially heterogeneous resources. Because of the fact that some of the resources are mutually exclusively allocated, like some experimental devices such as a telescope or a wind tunnel, the demand for a generic resource reservation mechanism, comparable to a reservation schedule of a video conferencing room, was born. The term advance reservation denotes the ability to specify a guaranteed resource behavior together with a time interval during which the defined service has to be provided. Resources offering an advance reservation interface are capable of delivering a specific quality of service over a given time interval.

Advance reservation is an important feature in a Grid environment for several reasons. From a resource scheduling perspective, the ability to assume a predictable resource behavior offers the possibility to consider additional scheduling algorithms and policies. Snell et al. have shown that a meta-scheduler, which schedules a set of Grid resources, can improve the overall effectiveness of the Grid by requesting a deterministic resource in advance [120]. Hollingsworth and Maneewongvatana presented so called "imprecise calendars" which permit the efficient sharing of distributed resources with the ability to provide advance reservations to applications [70]. Their simulation studies demonstrate the benefit of their approach across a collection of workstation clusters. In Grid environments, this advantage is increased by the diversity of relevant resources.

From an application point of view, the ability to expect well defined resource capabilities in combination with the admission to network guarantees offers the opportunity to balance the load of the distributed applications efficiently on the set of resources. Application developers can reduce unproductive synchronization time and improve the overall throughput of their application.

In a distributed Grid environment the interconnecting network is a resource of major importance. It is therefore clear that any advance reservation architecture used by higher-level services to improve the economic use of the Grid resources should also apply its functionality to the network. While network Quality of Service (QoS) is a major research area of the networking community, most work has been done on immediate reservations, without offering the capability to reserve in advance. Driven by the demand of combining the schedule of a conferencing room with the ability to claim an existing network reservation, several studies were made to support advance reservation capabilities in networking environments [108, 136, 42, 30]. However, no study addressed the integration of those capabilities to a more general resource management framework of computational Grids.

Additionally, the specific network demand of Grid applications differs from the requirements addressed by general QoS research. While the main focus of QoS research is on real-time flows such as video-conferencing streams, the Grid introduces new high-end network applications [57], in which individual flows can have high bandwidth, from a few to many tens or hundreds of megabits per second (Mb/s). The network capability demand of Grid applications is dominated by a complex mixture of flows, varying from low bandwidth to high bandwidth and from delay sensitive to delay invariant, which may also change their requirements dynamically throughout their lifetime. Those update events can be triggered by the user using teleimmersion devices, by sensors gathering peaks of data, by a superscheduler which is balancing the load of a distributed application, or simply by a state change of the dynamic Grid environment. Furthermore, as network QoS should be accessible through a common advance reservation API which is used for all reservation requests on Grid resources, computational Grids require a slightly different QoS instantiation than it is addressed by the research community so far.

The actual provision of network Quality of Service can be accomplished by a variety of

mechanisms. However, the deployment of those mechanisms in the current structure of the Internet is rather rare. Data link layer mechanisms provided by technologies such as ATM assume a specific homogeneous infrastructure between two end-systems. Ferguson and Huston state that if ATM is not pervasively deployed end-to-end in the data path, efforts to deliver QoS using ATM can even be counterproductive [41]. Because of the mass market development of network interface cards which focused on Ethernet technologies, ATM is not likely to be deployed in an end-to-end fashion and is thus not a considerable technique in this context.

IP-based QoS mechanisms do not depend on a homogeneous infrastructure. Instead, they rely on the capability of each router to treat packets differently. To build end-to-end guarantees, a trade off between the granularity of packet differentiation and the associated overhead has to be made. While a flow-based packet differentiation facilitates the provision of strong service parameters such as a boundary for the packet delay, it does not scale for complex networks. On the other hand, the differentiation between traffic aggregates allows an efficient provision of QoS, but the achievable service parameters remain difficult for complex aggregates. Traffic engineering mechanisms extend the capabilities of aggregate-based provisioning of QoS by the ability to control the composition of aggregates.

An important aspect for building network services based on aggregates is the dynamic assignment of flows to aggregates. A specific management entity, called a bandwidth broker, handles this dynamic mapping for a single trust domain. From a Grid perspective, bandwidth brokers can be viewed as resource management systems for the “network resource”. The integration of a bandwidth broker into a general resource management framework of computational Grids is therefore a natural extension.

The provision of end-to-end guarantees for Grid applications is a complicated task. The heterogeneous and dynamic nature of the Grid makes it hard to create an environment that follows a homogeneous QoS concept. Bandwidth brokers can be used to intermediate between the evolving Grid resource management framework and the use of lower level QoS concepts. By encapsulating the underlying QoS mechanism, the provision of end-to-end guarantees only depends on the ability to offer the requested service capabilities, and not on the deployment of a particular mechanism.

However, the heterogeneous requirements of Grid applications with their challenging network demand need additional studies about the underlying QoS concepts. The achievement of the required throughput does not only depend on the lower-level capabilities of the network, but also on higher-level protocols. The use of the elastic TCP protocol, whose self-clocking feature dominates the transmission in many scenarios, typically requires advanced tuning algorithms [90, 133, 126] in addition to any guarantees of the underlying networking infrastructure. The related research thread covers not only the establishment of advanced network services, but also techniques which support the applications in using the offered services efficiently.

This dissertation evolves a flexible bandwidth broker architecture that addresses the particular requirements of emerging Grid applications under two major constraints:

- The bandwidth broker incorporates the assignable capabilities of the Grid resource "network" into a Grid resource management framework; and
- all service guarantees are established by standard IP-based QoS building-blocks.

Chapter 2

Computational Grids

This chapter reviews the concepts of computational Grids and lists specific network requirements for their emerging applications.

2.1 Future Problem Solving Infrastructures

Large-scale scientific research and engineering often relies on the collaborative use of distributed resources. An airplane designer might want to use a wind tunnel to evaluate the actual behavior of a particular part designed in a complex set of simulation runs on a supercomputer. Data gathered during the experiment is archived in data repositories which build the information clearinghouse for future post-processing and visualization tasks. Another example would be during the design process of a complex product, a group of engineers might want to establish a collaborative meeting in a distributed Virtual Reality (VR) environment at critical decision points, with the intention of discussing the different design options with the ability to actually use the VR environment to virtually perform the design actions on the virtual object in order to get feedback information about the impact of currently discussed process.

The fundamental idea of a computational Grid is to facilitate the routine interaction of those type of advanced problem-solving tools. A computational Grid uses high-speed networks to link people with computers, databases, and other devices. This subsection clarifies the use of the term computational Grid and gives an overview about the challenges associated with this infrastructure.

According to Foster and Kesselman the definition of a computational Grid is as follows [52]:

Definition 1 *"A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities."*

In a more recent article, Foster, Kesselman, and Tuecke emphasize the nature of the Grid with their description of the specific problem that underlies the Grid concept [55]:

"The real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations".

Definition 2 *A virtual organization is a resource sharing infrastructure built by a set of individuals and/or institutions who agreed on sharing rules regulating who is allowed to access which resource.*

In this sense, the fundamental goal of building computational Grids is to enable the establishment of virtual organizations which facilitate the routine interaction of scarce high-performance devices without the need to replicate those expensive devices at each institution. Grids can be viewed as a middleware infrastructure serving scientists and engineers of a specific community. An important aspect here is that by referring to the construction of virtual organizations the underlying infrastructure does not necessarily consist of dedicated resources, but of protocols and services which can be used to access existing resources. Therefore, resources and individuals might be part of multiple Grids.

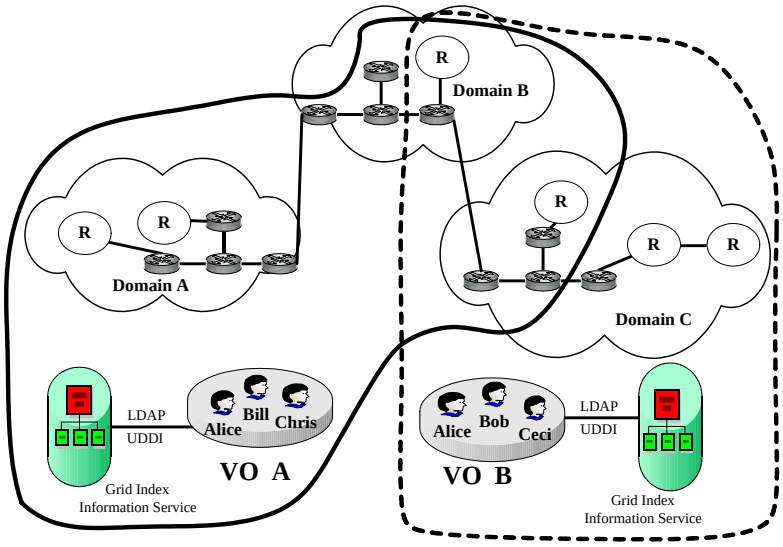

Figure 2.1: Two virtual organizations are built by resource sharing rules and a common name space. The network is part of these sharing rules.

Figure 2.1 illustrates an environment where two computational Grids are built by two different, but overlapping virtual organizations. Some resources are part of both Grids. It is

important to note that the network, if viewed as a Grid resource, is also part of multiple Grids. Here, the environment consists of three institutions, i.e administrative domains.

Of course, the middleware used to provide the related services is quite challenging to build. Structuring and standardizing the protocols and services is necessary to succeed in building a flexible Grid infrastructure for many heterogeneous communities.

In some sense, the Grid can be compared with the World Wide Web. While the fundamental goal of the World Wide Web is to offer location independent access to information resources by using a simple protocol, a common name space (the Universal Resource Locator), the integration of the name space into a standardized hypertext language, and a graphical user interface supporting these hypertext documents, the Grid is about to offer individuals and institutions the opportunity to build virtual organizations which facilitates the access to the problem solving resources of the community. The middleware infrastructure of a computational Grid provides the required set of services, such as security, information, resource management, data access, and event notification.

The application scenarios of a Grid environment are multifarious. Foster and Kesselman identify five major application classes for computational Grids [52]:

- Distributed supercomputing applications use the Grid infrastructure to aggregate computational resources to address very large problems such as the grand challenges [134, 73], i.e. problems that cannot be handled on a single system. Applications of this class are also known as metacomputing applications and typically also have challenging communication demands. While this scenario has been the focus of research for many years, any actual “metacomputing” service strongly depends on the capabilities of the available resources description languages and on advanced scheduling features such as reservation and coordinated allocation (co-allocation) of multiple resources.

- High-throughput computing applications use the Grid to schedule large numbers of loosely-coupled or independent tasks, with the goal of putting unused processor cycles to work. Examples include the use of multiple distributed workstations to solve hard cryptographic or complex design problems. While the intra-application communication requirement can typically be neglected, the staging of executables and data are an important issue in this scenario. To maximize the use of computing resources, executables and data should be reliably transported in a predictable manner.

- On-demand computing applications have challenging demands which cannot be fulfilled by the exclusive use of existing local resources. The Grid is used to meet those emerging short-term requirements by enabling the use of advanced remote capabilities to solve the problem. An example scenario for this type of application is the use of a local VR-device which is served by a remote supercomputer application, which transforms experimental raw data into the VR-environment in real time. This

facilitates the propagation of feedback information to the scientists controlling the experiment [103] in real time.

- Data-intensive computing applications use geographically-distributed data repositories, digital libraries, and databases to synthesize new information. The synthesis process is often computationally and communication intensive. An example scenario is addressed by the GriPhyN-project [60] which is facilitating a worldwide scientific community to extract small signals from enormous backgrounds via computationally demanding analyses of datasets, originating from large scale experiments. A reliable predictable fast transport of large amounts of data is necessary for this type of application.

- Collaborative computing applications are primarily concerned about enabling and enhancing human-to-human interactions. Shared access to data and computational resources is one of their major objectives. Examples include collaborative design activities and ”virtual worlds”. Because of the interactive character of this type of application, its resource demand is comparable to that of many real-time applications.

To schedule the resource usage efficiently and to serve applications based on their needs, service guarantees are important. Higher-level Grid services, such as a super-scheduler scheduling metacomputing applications on a set of computing resources, could increase their efficiency by using the assumption of a deterministic resource behavior at a given time interval. From the user perspective, the ability of accessing specific network service classes could allow them to balance their distributed application more efficiently, and thus to increase the overall throughput of the Grid by reducing the overhead.

While service guarantees are desirable by themselves, the use of resources which do not allow the assignment of multiple users in parallel results in the requirement of Grid middleware capable of accepting the related service requests in advance.

Scheduling services of a Grid could use existing advance reservation mechanisms to reduce the amount of unused resources. Additionally, a resource authorization service could benefit from the ability of specifying a time interval for a particular service request, because the cross-dependency between any remote data access, the authorization request to use the desired computing resource, and the availability of further analysis and management tools can be reflected by a given reservation schedule for each operation. Finally, collaborative environments are well known to rely on specific network capabilities. Any interactive audio and video communication is sensitive to the delay and the delay variance (jitter). A Grid resource management system could use an advance reservation feature by scheduling the end-resources together with the ability to claim the required network service.

Computational Grid applications extend the network QoS capabilities typically of interest, i.e. delay and jitter boundaries or bandwidth-guarantees for low-bandwidth micro-flows, by

the demand for a reliable guaranteed high-throughput end-to-end communication between resources. The Grid also requires focusing network QoS research on a complex mixture of flows, capable of supporting significant bandwidth guarantees to single streams.

It is important to note that network service requests are coupled with traffic producers and consumers. Hence, the reservation/allocation of a particular network service is combined with the reservation/allocation of further Grid resources. However, the coordinated parallel use of heterogeneous resources is still a major research field. Efficient resource selection mechanisms are needed as well as a reliable robust protocol for enabling the co-allocation of multiple resources. An advance reservation service for computational Grids should be able to control the access to the network premium service, but should also be capable of reserving a set of nodes on a Linux cluster.

A high-speed network infrastructure is of course a fundamental requirement for this environment, because the bandwidth demand of Grid applications can be significant. It is the deployment of very high speed networks over long distances which allows the distribution of resources in such a way that their actual geographical location is not necessarily a critical point, except for the delays introduced by the limit of the speed of light. The current technical network development promises the deployment of optical overprovisioned networks in the near future. However, the dynamic nature of the Grid has to address the question on how to serve applications with added-value network services, based on already deployed networking hardware as well as on possible future technologies. Thus, end-to-end network guarantees have to be provided by a flexible middleware layer which is capable of handling multiple types of network techniques.

2.2 Building Blocks for Computational Grids

The Grid will not be a monolithic, world-wide entity which offers its resources to all Grid users. Instead, there will be multiple virtual organizations, i.e. communities consisting of a set of individuals and institutions, building their specialized problem solving environment. The future infrastructure for scientist and engineers will consist of multiple computational Grids which, however, will share a common name space. It is likely to happen that selected services will be open for general use and some Grid individuals and/or organizations will have access to different computational Grids.

Therefore, Grid services should be standardized. Similarly, protocol bindings are required to assure the interoperability between the service requester and the service provider. This standardization task is one of the roles of the Global Grid Forum [59]. It is supposed to define standardized services and their interaction in such a way that components can be implemented that can be combined and used to build up a computational Grid. This subsection discusses the building blocks for setting up a Grid environment.

2.2.1 Core Services

Computational Grids are built with services and protocols that link together service provider and requester. Because of the complexity of the required middleware, services can be structured in fundamental low-level services, also called core services, and high-level services. While core services are essential for the basic operation of computational Grids, high-level services provide advanced features which are developed on top of the core services and used to provide more sophisticated solutions. We now describe the fundamental set of core services of a computational Grid.

Foster et. al. [55] relate this set of core services to the connectivity and to the resource layer of their Grid protocol architecture (refer to Figure 2.2).

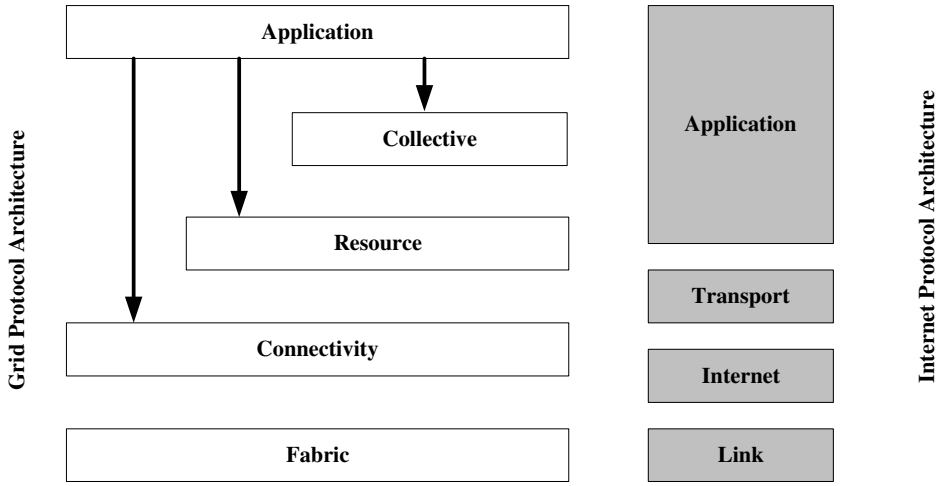

Figure 2.2: The layered Grid architecture and its relationship to the Internet protocol architecture [55].

Security Services Authentication and authorization are essential functions in any distributed environment. For Grid environments, however, the complexity of the security infrastructure is increased by the fact that a Grid combines resources which are maintained by different independent organizations. The problem here is that each institution might have defined its own security policy. Even the existing local security infrastructure might be established based on different authentication systems such as Kerberos [80]. Butler et. al. [18] identify the following requirements for Grid authentication schemes:

Single sign on The ability to create credentials based on a single authentication which can than be autonomously used for further authentication purposes during a specific time period enables a comfortable use of the Grid.

Delegation The ability of a subject to achieve a subset of rights of another subject facilitates the implementation of generic services and agents which act on behalf of the user, following a consistent authorization scheme.

Integration with local security solutions The ability to adopt from the global security scheme to the security system of the destination domain is essential to deploy Grid technologies into existing domains.

User-based trust relationships The ability to delegate rights requires the ability to trust entities, and to formulate policies describing the trust relation. As trust, in general, is not transitive, this mechanism allows the establishment of a transitive web of trust.

While single-sign on and the integration with local security solutions are fundamental Grid security service requirements, the need for delegation depends on the underlying concept of the site-integration. A trust model based on a strongly connected web of trust as used in the UNICORE project [109] would allow to avoid the general delegation. Because this concept can also be viewed as a user-based trust relation between the end-user and the UNICORE management system in general, it can also be associated with a limited delegation, where the management system receives the right to impersonate a single given request.

Information Services Because the Grid consists of a variety of services distributed over a set of institutions, the location of appropriate resource candidates is a major issue for the provision of any service. The dynamic state of the Grid, where the availability of resources varies over time, requires more complex resource location resolution than a common resource naming space would offer. Access to an information repository, storing information about properties and facilities of each transient Grid resource, is required. Examples for the information stored within this service are the current state of a resource, services it is providing, and protocols available to claim them. This all would allow a Grid application to select the appropriate resources for the desired service.

Resource Management Services Any service is offered by some hosting environment, i.e. by some set of resources. A Grid resource management service has to translate a given service request to an actual allocation of a set of resources. There are several levels of resource management services possible. The basic service must be capable of instantiating a single service request on a particular resource candidate. The service request itself must contain the required information about 'what' is requested, and attributes describing 'where' it is requested. Beside the support for the fundamental service specification, advance reservation mechanisms facilitate the specification of an additional constrain: 'when' the service is requested. Therefore, the resource management service embeds a unique interface to quality of service techniques for different types of resources.

The resource management service is coupled with other core services, like the security service which is required to assure the authenticity of all service allocations. It is used by higher-level services to provide more advanced capabilities such as to schedule the access to a wide variety of resources existing in many different security domains, or to support the simultaneous coordinated allocation of multiple resources.

2.2.2 High-Level Services

Since the Grid provides a problem-solving infrastructure for a particular community, its services are likely to be customized by the community to reflect their collaborative intention. While some higher-level services such as a super-scheduler which is scheduling a service class on a set of resource candidates will be of common interest for all Grids, others such as an automated interface to a particular experimental device will rather be very specific and thus not of interest for a broader community. This paragraph lists some high-level services which are typically of general interest.

Communication Services In a distributed environment, access to any service typically requires some communication. Service requests, however, are often connected to additional resources and services, such a data sets offered by some data repository, or the use of some potentially remote VR-device. A library providing a communication service which automatically uses the security service to authenticate and authorize the communication channels provides a convenient interface to all application developers. This software layer could easily be improved by automatically addressing relevant performance issues. In addition to automated network tuning, this step might include the integration of QoS mechanisms.

Instrumentation and Notification Services Grid applications should be capable of using a notification service for application related events such as system failures or state changes. An instrumentation service could help to receive important information from performance analysis tools and would allow to improve the economy of both the application and the Grid itself. Similarly, higher-level services, such as schedulers, should be capable of monitoring their resources, including convenient mechanisms to perform fault detection. A generic instrumentation and notification service allows applications to discover relevant information sources on the Grid. From an instrumentation perspective this service could link to performance analysis capabilities such as the Network Weather Service (NWS) [137]. An event notification service would connect event producer with event consumer [119].

Remote Data Access Services Since the Grid allows to decouple data resources from computing resources, remote data access is a major building block of a Grid. Access to the data produced by scientific data sources can be done by a whole community, using their accessible supercomputing facilities to process the data, and, potentially, locally installed Virtual Reality (VR) devices to visualize the experimental results.

Scheduling and Co-Allocation Services While a basic resource management service is concerned with providing an abstract resource interface for reservation and allocation of different resources, the Grid should facilitate the simultaneous and coordinated use of multiple resources. Whenever an end-user requests a service from two particular computers and the interconnecting network, three resource allocation steps with optional preceding reservation steps have to be performed. The provision of strict reliability semantics in the face of network, client, and server failures is an important aspect in this environment. Hence, a specific resource acquisition protocol has to be defined to deal with this issue.

An implementation of this co-reservation and co-allocation protocol could also serve users in such a way that it would schedule an abstract resource request to a set of possible resource candidates. These schedulers must be able to identify, authenticate, and authorize users according to local policy before attempting to schedule and allocate any resources. On the other hand, users must be able to identify, and authenticate the scheduler they connect to. Consequently, Grid schedulers will need to be able to securely act on behalf of the user as they attempt to schedule resources on her behalf. The user should be able to delegate to the scheduler some subset of her rights, including possibly the right to further delegate those rights.

2.3 Grid Applications

The classification of Grid applications given by Foster and Kesselman covers a wide range of scenarios [52]. This section refines their classification with respect to the specific requirements for network Quality of Service.

2.3.1 Distributed Supercomputing

Low-latency and high-throughput communication are performance-critical in most parallel programming environments, regardless whether the application is executing on a tightly coupled computer system using specialized interconnect hardware, or on a loosely coupled cluster connected through standard Internet technologies. Any synchronization of computation and any data propagation has to be done efficiently in order not to waste expensive and scarce supercomputing resources by waiting on incoming messages. This is especially

true when either the scheduling policy or the end-system scheduler does not allow gang scheduling [40] and thus would not be able to temporarily assign the unused resource to another application. According to Messina a range of techniques have to be developed to tackle this issue [92].

While the deployment of new latency-tolerant algorithms is still crucial, adaptive techniques that can react to the varying availability of Grid resources are another possible solution for this problem. Hence, a Grid instrumentation and notification service used by a Grid management system facilitates the development of adaptive applications. Combining this feature with the availability of advance reservation mechanisms, state changes could be propagated in advance, to relax the time constrain for adaption. Besides these advanced concepts, many distributed applications will still require challenging tightly bounded communication capabilities to perform well. Adaptation helps to balance the load based on the current state of the Grid, but there will still be latency boundaries and bandwidth expectation which must be fulfilled.

Whenever the communication libraries have to use standard network protocol stacks such as TCP, latency and bandwidth become an even more important issue. It is important to note that from a distributed application point of view the term latency describes the time it needs to propagate a short—typically 0 bytes—fixed length message from node A to node B. Hence, this view of latency also includes the time it takes to process and propagate the full protocol stack. The application itself is interested in the time it takes from writing the first byte of the message to receiving the last byte. Depending on the message length, specific properties of the underlying transport protocol come into place which even might become the dominating throughput limiting factors.

The traffic profile of distributed supercomputing applications itself can be viewed as bursty. This is because the processing of those applications is divided into computation and communication phases [93]. Figure 2.3 illustrates this behavior. Here, an application is writing 128 kilobytes (KB) blocks of data to the 384 KB socket buffer, with some occasional bursts of 256 KB blocks. In this scenario, TCP writes each block of data at link speed.

Though latency-tolerant applications would allow some asynchronicity, the fundamental burstiness would still exist and could cause an increase in latency which can not be absorbed by the algorithm.

Figure 2.4 illustrates the behavior of the same application in a congested network. Note that the trace was taken in a local area network in which the round-trip time was of the order of a single millisecond. TCP was therefore able to recover more frequently then it would do in a wide area environment. What is shown, however, is the significant impact of the congestion on the message delay. Whenever a timeout occurs, the delay depends on TCPs internal timers and is typically at least several hundreds milliseconds. What is also shown is TCP's slow start behavior. Whenever the protocol stack recovers from a timeout, it limits the transmission speed and doubles it after every received acknowledgment. In a

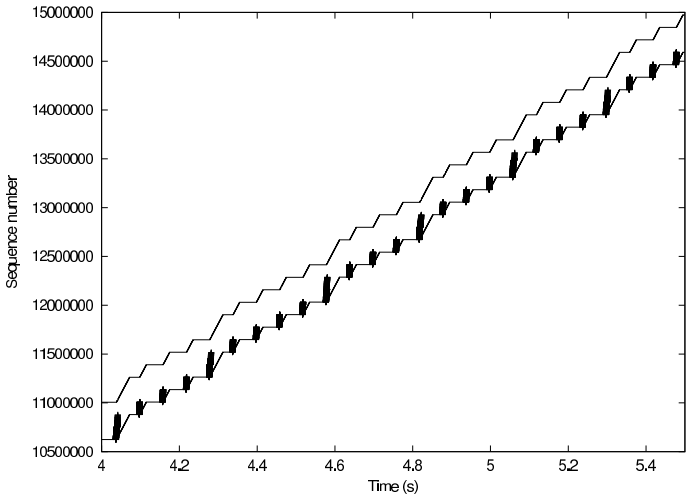

Figure 2.3: TCP sequence numbers shown over the time within the boundaries of TCP's sliding window. The slope indicates the throughput. The traced flow, given by the thick black line, is significantly bursty. The bottom line lists the acknowledgments over the time and is thus a representation of the network behavior.

wide area link, we would see a different slope (which represents the transfer rate). We will discuss the underlying mechanisms of TCP in more detail in Section 4.3.1.

From an MPI application point of view, any timeout should be avoided. Even a single packet drop causes TCP to react on this situation by a decrease of throughput. We therefore derive the demand for a service class which is capable of serving the applied traffic profile without any timeouts. A Guaranteed Rate (GR) service which is capable of serving the injected traffic without any packet loss and also providing delay boundaries, can provide an efficient load balancing of the application and is therefore a desirable service for distributed supercomputer applications. Unfortunately, the typical traffic profile of this type of applications does not fit very well to a GR service. Bandwidth is only claimed when the entities communicate. Hence, the particular challenge associated with distributed supercomputing applications is caused by the inherent burstiness.

In addition to the particular intra-application QoS demand, distributed supercomputing applications are coupled with the demand for a coordinated allocation of the related computing resources. This coordinated allocation step is often combined with an application level barrier which assures that the application startup of all parts is synchronized [28]. A consequence of this scenario is that any application startup can be delayed by a single resource. A typical example for this situation is an unfinished staging operation. While the other resources are already assigned to the application, the barrier prevents their actual use.

In an environment where advance reservation mechanisms are widely deployed, any scheduler for distributed supercomputing applications could cost effectively acquire a network service which guarantees the successful termination of the staging operation when the re-

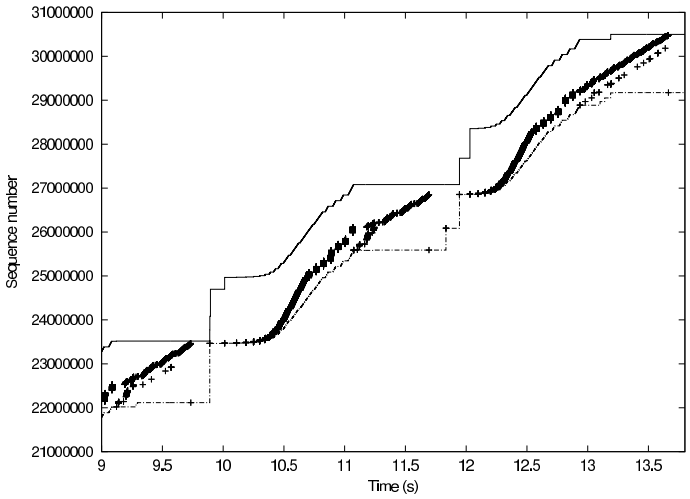

Figure 2.4: TCP sequence numbers shown over the time within the boundaries of TCP's sliding window. The slope indicates the throughput. The traced flow originated from the same application as illustrated in Figure 2.3. In this scenario, however, the network is congested. We see several packets retransmissions indicated by a "+" and two timeouts.

source becomes ready for use. Combining this with reservation bookkeeping mechanisms, the scheduler could optimize the decision process for finding appropriate resources and could avoid unnecessary blocking times due to missing staging data.

Thus, deadline-staging operations, i.e. the ability of a network service to guarantee the successful transfer of a given amount of data at a particular end-time, is an important requirement for this type of application. The provision of network QoS in Grid environments should therefore address this specific service demand.

2.3.2 High-throughput Computing

While distributed supercomputing applications try to couple multiple end-systems to achieve the computational capabilities of a superior computer, high-throughput computing (HTC) applications intend to gain additional computational capabilities over a long period of time. Both types of applications try to improve their service, but focus on a different time-scale. Another distinction between metacomputing and HTC is that the intra-application communication of HTC applications typically does not have the fine grained communication profile of metacomputing applications. Hence, it does not primarily require any special network service. The related pre- and post-processing effort might be significant though. Consider the transfer of large executables, or significant input or output data. Whenever this is the case, the economy of using a remote resource is affected by the time it takes to perform the staging operations. Of course, improving the efficiency

of the related transport operations does have an impact on the network load and, using standard Internet best-effort techniques, can thus influence the network service received by other applications. Hence, deadline-staging operations, are a useful requirement for high-throughput applications as well.

Another important aspect in the context of high-throughput computing applications is the implementation of a checkpoint service [86]. A checkpoint is a snapshot of a running program which can be used to restart the execution at a later time. Checkpointing itself is quite challenging, especially for parallel applications. Besides of the technical issue, checkpointing can also involve the propagation of a large checkpoint file. Depending on the location of this file, whether it is on a local or remote disk, checkpointing might also require the ability to efficiently transport the data over the network. By adding the ability of a GR service, i.e. an assigned portion of bandwidth available to the application, checkpointing operations can be efficiently placed in the processing of a HTC application.

Compared to the networking requirements of distributed supercomputing applications, HTC does not rely on added-value services. However, the ability to use services which were originally motivated by distributed supercomputing applications offers the opportunity to improve the overall effectiveness of a HTC environment.

2.3.3 On-demand Computing

The dynamic character of on-demand computing applications results in a high variance of their related network service requirements. Experimental feedback mechanisms often require to couple devices such as a Magnetic Resonance Imaging (MRI) device with a supercomputer application [37] which performs a tomographical post-processing of the experiment in real-time. While the time when this type of application claims the specific real-time capabilities is often known in advance, the demand of a supernova detection application is driven by cosmic events which require the coordinated ad-hoc usage of additional supercomputing resources [75]. In this scenario, telescopes permanently scan the sky and produce large amounts of raw data which is post-processed by computing facilities. The results can be mapped to those of complex simulation runs to filter out candidate supernovas for further observations. Whenever a promising supernova candidate has been found, advanced additional devices are requested. Examples are the reservation of the Hubble space telescope or the request for additional computing power to get a more accurate simulation result, to re-validate the experimental results in real-time and thus to avoid the waste of an extremely scarce resource, like the Hubble space telescope.

The key issue here is that in contrast to an MRI-application, the schedule of cosmical event is not predictable. Whenever one candidate has been selected, the application has to be finished within a given time interval, otherwise the event might no longer be observable. The amount of data, the location of the sensors, and the resource capable of providing the

computational service in time might vary. Thus, the mapping to resource providers should consider all these constraints. Any reservation capability would help in fulfilling this task. The demand of networking capabilities in this environment is of course significant as the pre-selection of candidates has to be accomplished within 24 hours. Exposures are done ever few hundred seconds and require the transport of data in the order of one gigabyte. Hence, sustained data transfer rates of up to 50 megabits per second (Mb/s) have to be accomplished [87].

The variety of network QoS requirements for on-demand computing is large. Applications such as the MRI-application often rely on the ability to deliver feedback information in real-time, and potentially are able to remotely steer the experiment. In both cases, the communication is quite delay sensitive. On the other hand, on-demand computing applications also rely on a coordinated allocation of multiple resources. Deadline file transfer is a possible requirement too. Applications such as the supernova detection rely on the ability to transfer data within a reasonable time scale. While tuning based approaches are one way to address this demand, the particular sparsity of resources such as the Hubble space telescope indicate that a GR service is what is actually needed by this type of application.

2.3.4 Data-intensive Computing

Scientific instruments and supercomputer simulations generate large amounts of data: tens of terabytes today, petabytes within a few years. Remote interactive exploration of such datasets requires that the conventional visualization pipeline can be decomposed across multiple resources. A realistic configuration might involve moving data at hundreds or thousands of Mb/s to a data analysis and rendering engine which then generates and streams real-time Moving Picture Experts Group 4 (MPEG-4) encoded video to remote client(s), with control information flowing in the other direction. QoS parameters of particular interest for this class of application include bandwidth, latency and jitter; resources involved in delivering this QoS include storage, network, computing, and visualization resources.

In other settings, large datasets are not visualized remotely but instead are transferred in part or in their entirety to remote sites for storage and/or analysis. The need to coordinate the use of other resources with the completion of these multi-gigabyte or terabyte transfers leads to a need for QoS guarantees of the form “data delivered by deadline” rather than instantaneous bandwidth. Notice that achieving this goal requires the scheduling of storage systems and computing capabilities as well as networks. Otherwise the application would not be able to actually achieve the required transfer rates.

Emerging file transfer applications supporting these challenging characteristics follow a pragmatic but necessary principle which we will call the “easy-to-deploy” paradigm. This

paradigm is derived by the fact that the fundamental nature of computational Grids is extremely heterogeneous. It is thus important to build the middleware on top of a commodity fabric, i.e. on top of commodity operating systems, network devices, and standard Internet protocols. Hence, instead of relying on capabilities of a new transport protocol or on any major update of existing TCP implementations—which both should be evaluated in parallel—they rely on no very specific capability of the TCP protocol variants and can thus be used on most operating systems. To achieve the required transfer rates, existing tools use parallel sockets with efficient load balancing algorithms to reduce the impact of TCP's congestion control mechanisms. This, however, is contrary to the common desire to deploy congestion control mechanisms in the Internet, i.e. to avoid a networking collapse caused by congestion. A service differentiation/isolation and a better control of the actual resource usage is a possible solution for this contradiction.

2.3.5 Collaborative Computing

High-end collaborative work environments involve immersive virtual reality systems, high-resolution displays, connections among many sites, and multiple interaction modalities including audio, video, floor control, tracking, and data exchange. For example, the NCSA Alliance "Access Grid" currently connects about 15 sites via multiple audio, video, and control streams, with the audio streams especially vulnerable to loss. Such applications require QoS mechanisms that allow the distinct characteristics of these different flows to be represented and managed [29].

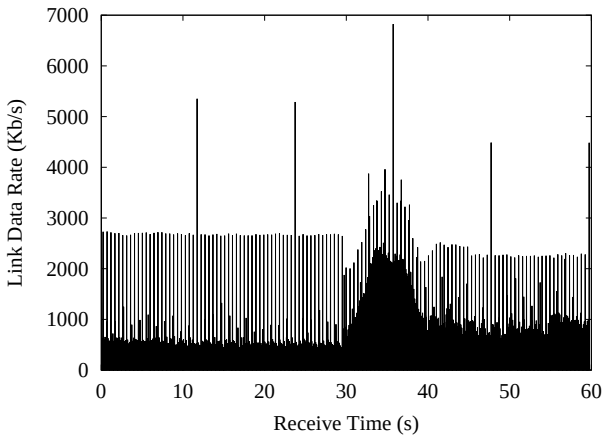

Figure 2.5: Traffic profile generated by an MPEG-4 encoded TV news sequence. The scenario can be compared to the traffic demand of a video-conferencing application.

The actual requirements obviously depend on the underlying cooperative technique. For video-conferencing data, publicly available video traces [46] can give a good estimate of the requirements. Figure 2.5, for example, lists the sequence of frames transmitted in a

television news sequence produced by ARD, a German broadcast television station. We assume that the scene content is quite close the content of video-conferencing data. The sequence was encoded by an MPEG-4 encoder with 25 frames per second and a Group of Pictures (GOP) pattern leading to the following flow characteristics: The minimum frame size is 123 B, the maximum frame size 17 KB, the mean rate is 0.722 Mb/s and the peak rate 3.411 Mb/s, leading to a peak to mean rate ratio of 4.72.

For illustrative purposes, we also examine a teleimmersion example in more detail. Consider two or more users at geographically separate locations who are collaboratively exploring a three-dimensional visualization of experimental data. As in other telecollaboration systems, we have a number of streams with fairly constant rate and low to moderate bandwidth: audio and video streams for communication, and jitter- and latency-sensitive streams for the tracking data indicating user movements in the virtual space. In addition, we have a number of streams with higher bandwidth and often variable rates, used for visualization data and (in some cases) database updates. Visualization data is calculated from the data set, and a representation of it, perhaps a set of polygons for rendering, is transmitted [49]. The actual amount of data being sent depends on both the data being visualized and user actions, which may include zooming and movement in space and time. In addition, contention for shared resources such as disks and Central Processing Units (CPUs) can also affect the transmission rate. Database updates or reads occur if users modify or annotate data and can require the ad hoc propagation of substantial changes.

Beside the influence of those intra-application parameters it is important to note that there are also several external parameters affecting the actual transmission rate. Disk and CPU competition are two examples for external parameters.

Characteristics such as these place substantial demands on both network infrastructure and applications. For example, consider a situation in which several teleimmersion sessions are in operation simultaneously, while other groups are concurrently attempting to perform high-speed bulk-data transfers over the same network infrastructure, perhaps to stage data required for an experiment later in the day. With today's protocols and services, no group would obtain acceptable service. A single teleimmersion application has challenging QoS demands.

Table 2.1 lists the various flows of a future teleimmersion application together with their networking demand.

As a consequence of this, both resource providers and consumers should be able to specify and implement flexible resource allocation policies. For example, in the situation just noted, resource providers might allocate resources to different teleimmersion sessions and bulk-data transfers differentially. For example, teleimmersion session *A* might have priority, while sessions *B* and *C* would be guaranteed some minimum service. Bulk-data transfers *D* and *E* would have lowest instantaneous priority but would be guaranteed service in terms of another "terabytes per hour" metric.

	Latency	Bandwidth	Reliable	Multicast	Security	Streaming	Dyn QoS
Control	$<$ 30 ms	64 Kb/s	Yes	No	High	No	Low
Text	$<$ 100 ms	64 Kb/s	Yes	No	Medium	No	Low
Audio	$<$ 30 ms	128 Kb/s	No	Yes	Medium	Yes	Medium
Video	$<$ 100 ms	5000 Kb/s	No	Yes	Low	Yes	Medium
Tracking	$<$ 10 ms	128 Kb/s	No	Yes	Low	Yes	Medium
Database	$<$100 ms	$>$ 1 Gb/s	Yes	Maybe	Medium	No	High
Simulation	$<$ 30 ms	$>$ 1 Gb/s	Mixed	Maybe	Medium	Maybe	High
Haptic	$<$ 10 ms	$>$ 1 Mb/s	Mixed	Maybe	Low	Maybe	High
Rendering	$<$ 30 ms	$>$ 1 Gb/s	No	Maybe	Low	Maybe	Medium

Table 2.1: Networking flows and their QoS requirements of teleimmersion applications [29]. Especially Haptic and Tracking flows require access to a low-delay service.

The above listed demand results in the requirement to build a QoS mechanism which supports the following scenarios:

- Individual flows can have high bandwidth, from a few megabits per second (Mb/s) to many tens of Mb/s.
- There are complex mixes of flows, from low bandwidth to high bandwidth, and low latency to high latency. Access to a Premium service which is addressing the challenging latency demand is essential for the support of distributed Haptic or Tracking streams.
- Some flows dynamically change their requirements throughout their lifetime, depending on the user's action.

2.4 Conclusion

This chapter discussed the structure of a future problem solving infrastructure: computational Grids. The deployment of this infrastructure requires sophisticated middleware, including a specific resource management framework which provides a unique interface to the services built on top of the Grid resources. One task of this resource management framework is to support the coordinated allocation of multiple resources within a single service request. In this context, advance reservation capabilities are an important feature to support this task in a Grid environment.

The network is a Grid resource of major importance. Whenever it is capable of providing specific services, the related access mechanisms should be integrated into the general resource management framework. The specific network demand of Grid applications differs from the requirements addressed by general QoS research. This chapter listed the particular requirements and motivated the related services, namely a Premium service offering a low-delay virtual leased line and a Guaranteed Rate service.

Chapter 3

IP-based Quality of Service

Computational Grids provide services on a set of distributed and heterogeneous resources. The transformation of a given request to a particular set of resources can be done more precisely when the request is claiming well defined and guaranteed resource capabilities, i.e. Quality of Service (QoS). Considering the existence of QoS mechanisms in a Grid infrastructure, efficient bookkeeping mechanisms of resource management systems could be evolved to efficiently select those resource candidates which are actually capable of fulfilling the service request. The benefit of Quality of Service for Grid environments is increased by the fact that computational Grids are supposed to support collaborative environments which are strongly related to classical network QoS research. Typically, Grid applications will require the use of multiple resources and, especially, the network in between, and thus want to rely on QoS capabilities.

This chapter is structured as follows. First, two existing frameworks for the provision of network QoS are described: the Integrated Services (IS) and the Differentiated Services (DS) architecture. Based on the concept of a bandwidth broker, a middleware service which is responsible for controlling the access to services classes in a DS domain, it motivates the appropriateness of the DS architecture in the context of a Grid resource management system. The capabilities of a DS environment are then extended by a flexible switching technique which adds the ability to perform traffic engineering. Finally, this chapter describes the network calculus, an analytical model for describing the worst-case behavior of the network.

3.1 Overview

There are two basic approaches to providing network Quality of Service (QoS): reservation-based and adaptation-based [43, 83]. Applications using the reservation-based

approach usually rely on specific capabilities of the underlying network infrastructure. They claim the demanded capabilities during connection establishment and typically do not change their requirements subsequently; the QoS system in turn guarantees that (modulo system failures or preemptions) the reservation will not change during the lifetime of the application. Here, network QoS refers to the ability of the network to handle specific packets in such a way that the related flows receive a specific type of guaranteed service, i.e. leaving the current best-effort service. With respect to a Grid resource management framework the reservation-based approach nicely fits into a common resource management infrastructure where reservations can be placed on a broad set of resources, including the network.

On the other hand, applications that use the adaptation-based approach do not make a reservation but adapt to the network conditions at hand by responding to some form of feedback, whether explicit (notification of network conditions) or implicit (noticing that the achieved bandwidth is low). Instrumentation and notification are the key issues in this approach. Adaptation may occur when the application detects a problem or when the application is notified that a problem may exist [85, 132]. In this context, network QoS refers to the ability of an application to adapt its network resource usage to the actual state of the network in such a way that the fundamental functions of the application are still performed.

The reservation-based approach relies on the ability of the network to provide a specific type of service. Service levels can be characterized by the following QoS-related parameters [19]:

Bandwidth The rate at which packets of a flow must be carried by the network. The specification of bandwidth guarantees includes the assurance of a peak data rate, a sustained data rate, and a minimum data rate. The latter is important, as it assures the ability to use this capacity at any time. The sustained date rate can either be a statistical guarantee, or, in the more common case, equal to the minimum date rate. The peak data rate is often specified in term of bursts relatively to the sustained data rate [67].

Delay An upper boundary for the time it takes for send an MTU-sized packet (see below) to traverse the network from the source to the receiver (end-to-end delay). A more detailed definition of the Type-P-One-way-Delay can be found in [4]. For some applications it is useful to limit the delay for a full round-trip.

Jitter The variation of delay. Its formal representation is currently discussed by the Internet Engineering Task Force (IETF) as the ”instantaneous packet delay variation” [31] and is the difference of the delay experienced by subsequent packets on a one-way transit from source to destination.

Reliability This parameter specifies the percentage of lost packets, errors in the network due to non-failure events, the mean-time between failures, the mean-time to repair, and the availability as percent of the uptime.

Maximum Transfer Unit Because network traffic is typically not contiguous but separated in discrete packets of a maximum size, the maximum transfer unit is a service relevant parameter. First, it denotes a bound for the packet header overhead. Second, it indicates possible fragmentation delays for the application. Finally, it is relevant for packet scheduling as packets in transit are typically non-preemptable.

3.2 IP-based QoS Architectures

QoS in IP-based networks can be provided by a differentiated packet treatment. There are two basic approaches: one is differentiating the treatment of packets on a per-flow base and one is grouping packets to aggregates which are treated in a pre-specified way. This section summarizes these two techniques and extends the aggregate based approach by the ability to perform traffic shaping.

3.2.1 The Integrated Services Framework

The Internet Engineering Task Force (IETF) has specified the Integrated Services (IS) Framework [139, 15] with the goal to provide end-to-end QOS to applications. The basic framework is comprised of two elements:

- An extended service model which is also called the IS model.
- A reference implementation framework.

The extended service model consists of two particular real-time services which are built on specific capabilities provided by all networking devices. Namely, the model assumes that each network node is capable of differentiating packets based on their service-class and of a particular preferred treatment of those packets to influence the time-of-delivery under all conditions.

Figure 3.1 gives an overview about the reference implementation framework which was proposed in [15]. All routers capable of supporting the IS architecture should offer three components that implement the required traffic control within the network devices:

- *The Admission Control* procedure of a router decides whether or not to accept or reject the requested QoS of a new flow. It is important to note that the admission control test does not provide any specific QoS, instead it declines unacceptable request and polices whether a granted request is conforming to its specification.

- *The Packet Classifier* identifies the service class and maps it into the related treatment, i.e. to the related output queue of the *Packet Scheduler*. The service specific treatment includes accounting.
- *The Packet Scheduler* manages the packet forwarding as part of the output driver of a router. It uses a set of queues and perhaps other mechanisms such as timers to schedule the order of outgoing packets.

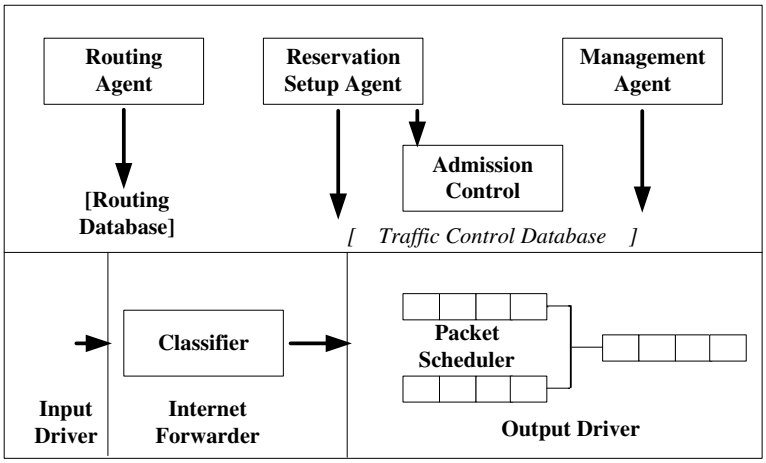

Figure 3.1: Integrated Services implementation reference model for routers. The functionality is divided into a control and a forwarding component. Services are provided by an router internal reservation agent which participates in the signaling process. The forwarding component is instantiating the service request by a packet classifier and a per-flow based packet scheduling strategy.

The implementation framework reflects the three key assumptions of the IS architecture [15]. The first assumption is that resources are managed explicitly to meet application requirements. This implies that ”resource reservation” and ”admission control” are key building blocks of the service. A signaling protocol, of course, is necessary to propose the service attributes to the network elements.

The second assumption is that networking devices must be able to store flow-specific states to establish different service level guarantees. To preserve IP robustness, this state can be soft state, i.e. it is maintained through periodic refreshes and timer-based removal.

The final assumption is that the framework does not rely on a specialized real-time infrastructure. Instead, a common infrastructure is used to support both non-real-time and real-time communication. This includes a unified protocol stack, e.g. IP.

The essence is a QoS infrastructure which offers services for individual flows which have applied for this in advance. Each router performs admission control to ensure that they only accept reservation requests when they do have sufficient local resources. Once an appropriate reservation has been installed in each router along a path of a flow, the particular treatment of the packets assures the level-of service during the life-time of the reservation. Note that the IS framework is a flow oriented mechanism.

The Integrated Services framework is built on two defined QoS control services:

- The Controlled-Load Service [138] provides flows with quality of service closely approximating the QoS that same flow would receive from an unloaded network element, but uses capacity (admission) control to assure that this service is received even when the network element is overloaded. Though it does not use strict boundaries of QoS parameters such as delay, it ensures that a very high percentage of the delivered packets will not experience a greatly exceeding transmission delay than the minimum transit delay due to propagation and router processing. Routers implementing the Controlled-Load Service must check for conformance of the related data flows to their reservation specification. Any non-conforming load must not be allowed to affect conforming traffic.

- The Guaranteed Service [117] provides a guaranteed network element behavior with a strict mathematical assurance of end-to-end datagram queuing delays and throughput. Each router is informed about the reservation in advance and associates a bandwidth R and a buffer space B for each flow registered in this service-class. The flow effectively sees a dedicated wire of bandwidth between source and receiver.

As stated above, the signaling of server requests is a key concept of the IS framework. The resource ReSerVation Protocol (RSVP) [16] was designed to enable the senders, receivers, and routers of communication sessions to communication which each other in order to setup the reservation described earlier. RSVP has several important features:

- RSVP is only a protocol for requesting services, i.e. a signaling protocol. It does not provide any built-in mechanism for routing or packet scheduling.

- The reservation specification is opaque to RSVP and usually is specified as described by the IS specification [139]. However, there are proposed extensions to this use for different environments [35, 8].

- RSVP requires the receiver to make the reservation and not the sender. The sender is required to provide the network with a traffic specification (TSpec). This model enables the receiver to modify the actual reservation to what it expects to be able to handle and facilitates a convenient multi-domain admission control.

- A fundamental idea of RSVP is the softness of the reservation status. The lifetime of a reservation is limited and is periodically refreshed by further RSVP messages. Thus, IP-robustness is reasonably preserved.

The basic operation of RSVP is divided into very few types of messages, all consisting of the same structure: a message header which specifies the message type and its length, and the RSVP objects. The fundamental two messages are the PATH and the RESV message:

- The sender applying for a particular service sends a PATH message to the receiver. This message contains a traffic characteristic (TSpec) object used in the admission control procedures. A detailed description of this specification is given in Section 3.4.

  While the TSpec object describes the original flow specific QoS requirements, the ADSpec is an optional object which includes both, parameters describing the properties of the data path and parameters required by specific QoS control services. It can be updated by each hop. The intention is that this object carries the information which is required by the receiver to determine the achievable level of end-to-end-QoS.

  The default general parameters includes the following fields, which are updated at each RSVP-capable router along the path:

  – Minimum path latency
  – Minimum of individual link bandwidth along the path
  – Hop count
  – Path maximum transmission unit

- Whenever the PATH message is received by the end-point, the receiver has to prove whether it accepts the service request. Whenever the response is positive, it sends an RESV message back to the source, containing a FlowSPec object which describes the level of service the receiver is willing to grant. The message is forwarded hop-by-hop taking the same but reverse path of the PATH message and causes each intermediate router to perform its own admission control check. When the check has been passed, the service request is instantiated and the RESV message is propagated to the next hop. Figure 3.2 illustrates this principle.

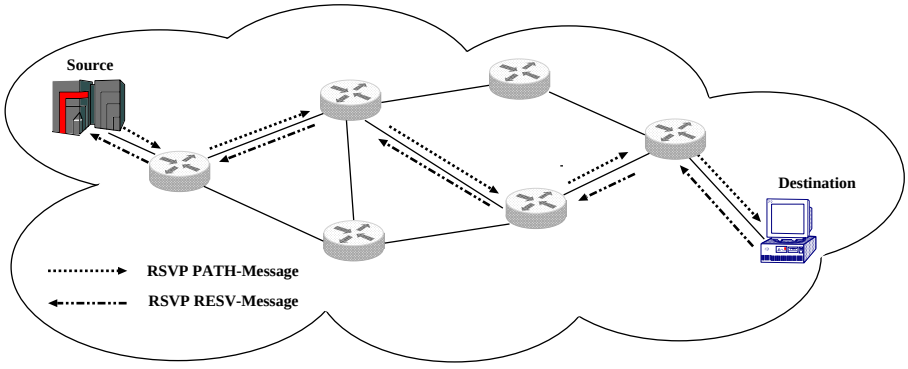

Figure 3.2: The basic principle of using RSVP for network reservations. The PATH message is propagated downstream. The receiver responds with a RESV message which is returned along the same path the PATH message took.

Deploying an environment capable of performing end-to-end guarantees using the IS framework would mean to install end-systems capable of performing RSVP signaling and routers, aware of RSVP messages ("PATH" and "RESV"). Whenever a service is instantiated by the receiver, each router would have to handle the specific flow in a particular manner. To perform this task, all packets would have to be parsed to select the packets of the particular flow and to schedule and treat them separately. A criticism of the IS approach is that this flow based concept would result into scalability and performance problems for large high-speed networks [88].

3.2.2 The Differentiated Services Architecture

The Differentiated Services architecture [11] is a reaction to the scalability problems of earlier per-flow-based network quality of service (QoS) architectures such as the above described Integrated Services architecture. While it is still following the fundamental principle of using a common infrastructure for both, non-real-time and real-time traffic, it leaves the reservation based approach of the Integrated Services architecture. In contrast to serving individual flows it focuses on defining the behavior of aggregates. Packets are identified by simple markings that indicate the applicable forwarding treatment. In the core of the network, routers need not to determine which flow a packet is part of, only which aggregate behavior they should apply.

The Differentiated Services architecture pushes complexity to the edges of the network: packets are marked to belong to an aggregate behavior either by applications or by edge routers. If edge routers mark packets, which is the more general solution, they may choose to do so on a per-flow basis or on any other criteria. In this scenario, of course, the question arises which packets will get marked. This is especially the case when the environment is dynamic, i.e. when varying flows should be able to use the available services. Here, a particular resource manager called a bandwidth broker comes into place.

Definition 3 *A bandwidth broker is a middleware service which controls and facilitates the dynamic access to network services of a particular administrative domain. Bandwidth brokers are also viewed as the Policy Decision Point (PDP) of the controlled domain.*

The concept of a bandwidth broker is typically associated with the Differentiated Services architecture. In this context, the task of a bandwidth broker is to control the configuration of the edge routers of a single DS domain. By performing a careful admission control bandwidth brokers are a fundamental building block for the provision of network services on top of DS aggregates. Figure 3.3 illustrates a DS domain controlled by a bandwidth broker.

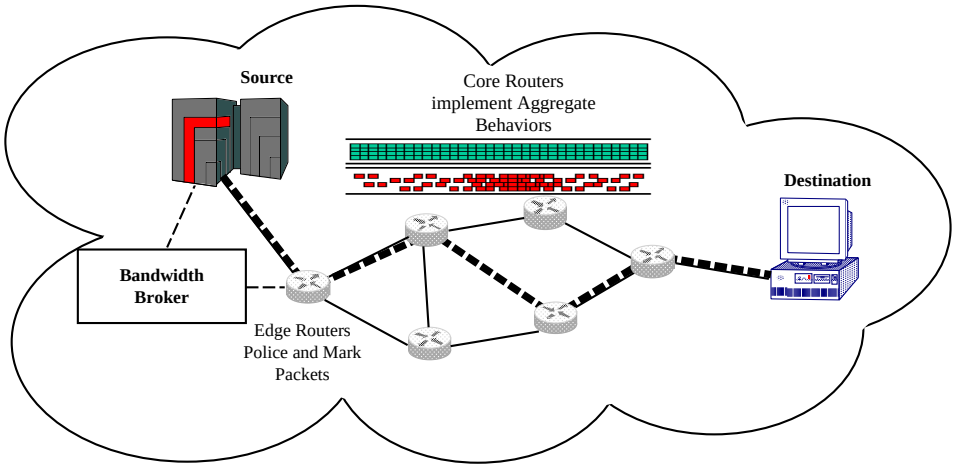

Figure 3.3: A network flow through a DS domains. The edge router is responsible for policing and marking packets. Core routers treat packets based on their aggregate. A bandwidth broker is used to control the configuration of the ingress router.

Packets may be marked by setting the first six bits of the type of service field of the IP-header [96] only when they are "within profile"—that is, when the sender is sending within predetermined limits, such as bandwidth or time of day. In contrast to this complexity in the edge routers, routers in the core of the network provide service based only on these markings. A particular marking on a packet, the Differentiated Services Coding Point (DSCP), indicates the applied per-hop behavior (PHB). Currently, the Internet Engineering Task Force's Differentiated Services Working Group has specified a small set of PHBs [77, 66] and their related marking.

The idea of the Expedited Forwarding (EF) PHB is the provision of a high-priority service. Its basic intention is to serve the aggregate with a configurable sustained rate, regardless of the amount of competing traffic in other aggregates. While the intention is actually quite clear, its formal specification has changed over time. It originally specified that the departure rate of a class of traffic from a router must equal or exceed a configured rate when measured over any time interval equal to or longer than the time it takes to send a packet of maximum size at the configured rate. Bennett et. al. [9] presented a family of networks which allows the construction of one network in this family which worst case end-to-end delay jitter is larger than any arbitrary value, even though the network is implementing the EF PHB. Having a low-delay Premium service build on top of the expedited forwarding PHB in mind, they proposed [22] a new definition as a forwarding treatment where the node offers a configurable so-called Packet Scale Rate Guarantee (PSRG).

Definition 4 *Let $a(j)$ denote the time of arrival of the last bit of packet j of the EF aggregate, $d(j)$ denote the time of departure of the last bit of packet j of the EF aggregate, and let $L(j)$ denote the length of this packet. A node offers to the EF aggregate a "packet scale rate guarantee R with latency E" if the j-th departure time satisfies the following condition for all $j \geq 0$:*

$$d(j) \leq F(j) + E$$

where $F(j)$ is defined iteratively by

$$F(0) = d(0) = 0 \tag{3.1}$$

$$\forall_{j>0} F(j) = max[a(j), min(d(j-1), f(j-1))] + \frac{L(j)}{R} \tag{3.2}$$

It is important to note that the PSRG is stronger than the assumptions of the Guaranteed Service for the Integrated Services architecture [45].

The intention of specifying per-hop behaviors for aggregates is always the establishment of services. Because the concept of DS is bound to domains in which the specific PHBs are applied, services are established by domains and are provided within domain boundaries. Nichols and Carpenter [95] formulated this particular property by the definition of a Per-Domain Behavior (PDB).

Definition 5 *The term Per-Domain Behavior (PDB) describes the expected treatment that an identifiable or target group of packets will receive from "edge-to-edge" of a DS domain. A particular Per-Hop Behavior (PHB) (or, if applicable, list of PHBs) and traffic conditioning requirements are associated with each PDB.*

The Premium service specification of the two-bit DS architecture for the Internet described in [97] addresses this intention. The fundamental idea of a Premium service is to provide a "virtual leased line" with little jitter and queuing delay. Premium service levels are specified in terms of a peak rate for a specific flow or an aggregation of flows. The admission control performed at the edge devices ensures that the use never exceeds this claimed peak rate. The reason for this policing function is the prevention of starvation of other traffic aggregates. In detail, the first-hop router filters the packets entering the network marks packets of a Premium flow.

End-to-end guarantees are limited to the minimum service level available in all transient domains. Whenever a single domain does not offer the appropriate service level, the application would not receive the required end-to-end guarantees. Hence, QoS deployment is likely to start with the highest level of service guarantees: the Premium service.

Beside of the Premium service a so called Olympic service [66] is proposed by the IETF to be based on the AF PHB group by extending it by means of a differentiated admission control and a class based over-provisioning. Three of the four currently defined classes

of the AF PHB group are used for such an Olympic service. The service differentiation between the three classes gold, silver, and bronze is proposed to be performed by the means of admission control, i.e. assigning only a light load to the gold class, a medium load to the silver class and a high load to the bronze class.

For managing such a configuration and to allow for an efficient implementation, a need for DS traffic engineering (TE) becomes obvious. Since DS alone can only control the overall network load of certain classes through admission control, but not the load on individual links, it cannot give meaningful deterministic nor probabilistic delay boundaries in this scenario.

A fundamental requirement to implement a per-hop behavior is to support specific well known features in each network device. Edge routers must classify, police, mark, and optionally shape the traffic. Core routers have to provide specific congestion avoidance and management mechanisms to establish the desired per-hop behavior.

Classifying, Policing and Packet Marking

The term "policing" denotes the ability to check whether a specific set of timely ordered incoming packets—$P = \{(p_i, t_i) | 1 \leq i \leq n\}$ where $t_0 < t_1 < t_2 ... < t_n$ indicates the arrival time of the relevant packets $p_0, p_1, p_2, ..., p_n$—is not exceeding a given traffic profile. A common technique for specifying a traffic profile uses the "Token Bucket Model". A token bucket is a non-negative counter which accumulates tokens at a constant rate r until the counter reaches a maximum capacity b, the token bucket depth. Upon packet arrival, the packet size in bytes is checked against the amount of tokens in the bucket. If this amount exceeds the packet size, the packet is treated as conforming and the actual amount of tokens is reduced by the size of the packet. If there is not a convenient amount of tokens in the bucket the packet is treated as exceeding its traffic profile. To clarify the use of a token bucket (r, b), consider that it is used for policing a flow in such a way that exceeding packets were dropped. Using this token bucket, one could assure that for any given time interval $[t_0, t_1]$ of length $T = t_1 - t_0$ the amount of data passing this policing function is not exceeding $rT + b$ Bytes.

An implementation of the policing function requires to actually classify incoming packets based on their characteristics, such as source and destination IP-address, Port numbers, or the Type of Service field of the IP-header. Once the decision whether a packet is relevant or not has been made, it is policed against the related traffic profile. Depending on the question whether the packet was conforming or not conforming, it receives a different treatment. A token bucket based marker which distinguishes between three different types of action is the Single Rate Three Color Marker (SRTCM) [67].

A common packet treatment used in this context is to assign a service specific DSCP value to the related IP packet. The action for packets exceeding the reserved rate could be to

either transmit them with a different marking, or to drop them explicitly.

Packet classification facilitates the transition from flows to aggregates. Policing is used for controlling access to aggregates, i.e. for admission control, and marking is the entry point to aggregates.

Congestion Management and Avoidance

While edge routers police and mark the incoming traffic, core routers implement the related aggregate behavior. Two major concepts exist to fulfill this tasks.

Congestion Management is the ability of a network device to perform a specific packet schedule when the output link is congested. The ideal approach is to associate a relative weight (or precedence) with each individual traffic flow, and at every router, segment each traffic flow into an individual First-In First-out (FIFO) queue, and configure the scheduler to service all queues in a bit-wise weighted round robin fashion. This is an instance of a Generalized Processor Sharing (GPS) [100] discipline.

A popular approximation to GPS is Weighted Fair Queuing (WFQ). WFQ offers dynamic, fair queuing that divides bandwidth across a set of queues of traffic based on weights. Given the weight of the queues, WFQ calculates the time the packet finishes service under the GPS scheme and ensures that packets are served in the order of the time they would finish their service in a GPS environment. Weighted queues can be established based on a value of the DSCP. That is, class-based WFQ is able to detect higher priority packets marked with a related DSCP and can schedule them in a well-defined manner. Class-based WFQ is conceptually very well suited for scheduling traffic with a marked precedence.

In periods of congestion each WFQ class is allocated a percentage of the output bandwidth equal to the weight of the related queue. For example, if an DSCP class is assigned a weight of 30, packets from this class will allocated at least 30 percent of the outgoing bandwidth during periods of congestion. It is important to note that WFQ only has an effect when there is congestion. When the interface is not congested, the treatment of packets is independent from their classification.

Bennett and Zhang have shown that WFQ is misbehaving in some environments [10]. However, these scenarios only apply when WFQ is used with many queues. Hence, when the service differentiation is rather moderate, WFQ is still a useful vehicle for congestion management.

Another popular packet scheduling mechanism is non-preemptive Priority Queuing (PQ). The concept is quite simple: packets of the particular priority queue will always be served first. However, as forwarding itself can only be done at link speed, the actual forwarding of a priority packet might be delayed by low-priority packets which are currently in transit,

i.e. which are currently located in the device queue. Priority Queuing is an appropriate mechanism for implementing the EF per-hop behavior.

While congestion management denotes a specific treatment of packets in case of an overloaded output link, congestion avoidance is used to implement strategies to reduce the likelihood of a congested output link. We now list popular congestion avoidance techniques:

Whenever there are more packets to be transmitted than the output link or the CPU of the router is actually capable to handle, queues are used to delay the transport of unprocessed packets without any packet loss. Because the size of those IP-layer queues is limited, a packet drop strategy is required. The simplest possible mechanism is to explicitly drop all packets, once a specific threshold has been reached. Often this threshold is equivalent to the length of the queue. Though tail drop is not actually an active congestion avoidance mechanism, it still represents a controlled reaction on congestion which is supposed to avoid this state in the near future.

Using tail drop is not a very successful mechanism to avoid congestion. The fundamental idea of many advanced congestion avoidance mechanism is linked to the capability of a TCP stream to react on an experienced packet drop with a reduction of the transmission rate. Whenever the corresponding sender recognizes that packets were dropped, it reduces its congestion window size, which immediately results in a smaller transmission rate. A more detailed presentation of this mechanism is listed in Section 4.3.1. For now it is reasonable to refer to the intention to avoid the experienced situation of congestion. It is important to note that this mechanism does not work for UDP flows. It requires an elastic protocol which reacts on dropped packets.

A common way to keep queues from overflowing is to use Random Early Detection (RED) [48]. In RED, the packet drop probability depends on the length of the IP-layer queue. Starting with a minimum threshold the algorithm linearly increases the drop probability up to a maximum queue length threshold. Once this threshold is exceeded, all packets get dropped. The slope between the two thresholds is a configuration parameter. Figure 3.4 explains the basic algorithm of RED.

Another popular, more advanced congestion avoidance algorithm is Weighted Random Early Detection (WRED). WRED is based on RED, but provides separate thresholds and weights for different classes of packets. Hence, it allows a service differentiation based on the DSCP value. By dropping lower-level service traffic more frequently than higher-level service traffic, periods of congestion can be resolved with less impact on higher-level service traffic.

Flows do normally not produce packets at a constant frequency. Even if the application itself is writing data at a constant frequency, protocol properties such as TCP's flow and congestion control or CPU competition influence the actual traffic profile. A result of this is that the burstiness of flows varies even at their source.

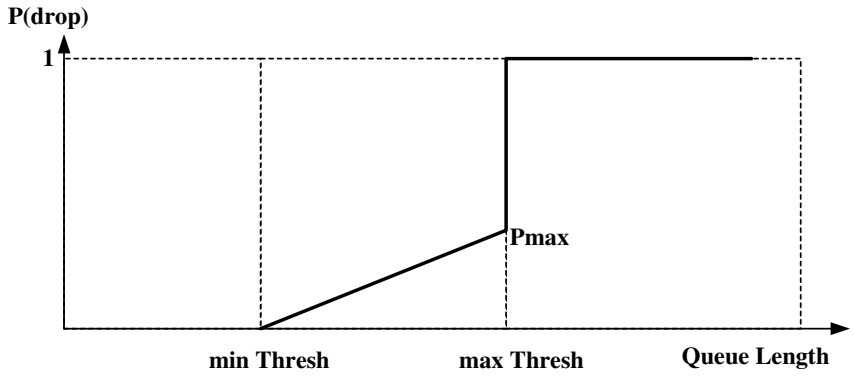

Figure 3.4: Drop probabilities of the Random Early Detection algorithm. Starting with a minimum threshold for the average queue length, the drop probability is increased linearly with the length of the average queue size up to a maximum threshold. The slope of the linear dependency is determined by a maximum drop probability P_{max}. Once the average queue length exceeds the maximum threshold, tail drop is performed, i.e. all incoming packets will be dropped.

In both architectures, the Integrated Services and the Differentiated Services, the incoming traffic is policed against a pre-defined arrival profile. Whenever a particular flow should be enforced to conform to the policed traffic profile, traffic shaping comes into place. Traffic shaping is forcing a particular series of packets to conform to a certain specified behavior.

Practically, traffic shaping can be implemented by a token bucket algorithm. It basically operates like a token bucket is operating when it is used for the policing function. Here, however, it uses additional queues (comparable to the holding queues describe in [97]) to hold non-conforming packets. When a packet arrives for transmission, the related amount of tokens is taken from the token bucket.

3.3 Multiprotocol Label Switching

The MultiProtocol Label Switching (MPLS) architecture [111, 110] is a label switching technique which is based on a functional decomposition of the network layer into a control component and a forwarding component. Flows of packets are grouped into disjoint subsets, which from a forwarding point of view are treated by the routers in the same way. This logical grouping of traffic with a like destination is called Forwarding Equivalence Classes. The members of these classes are identified by a common label. In contrast to other switching techniques such as ATM, MPLS does not rely on specific capabilities of the underlying link layer. Instead, it introduces the ability to add an additional label on top of the data link layer. It is therefore possible to deploy MPLS in heterogeneous environments which are built on different data link layer techniques.

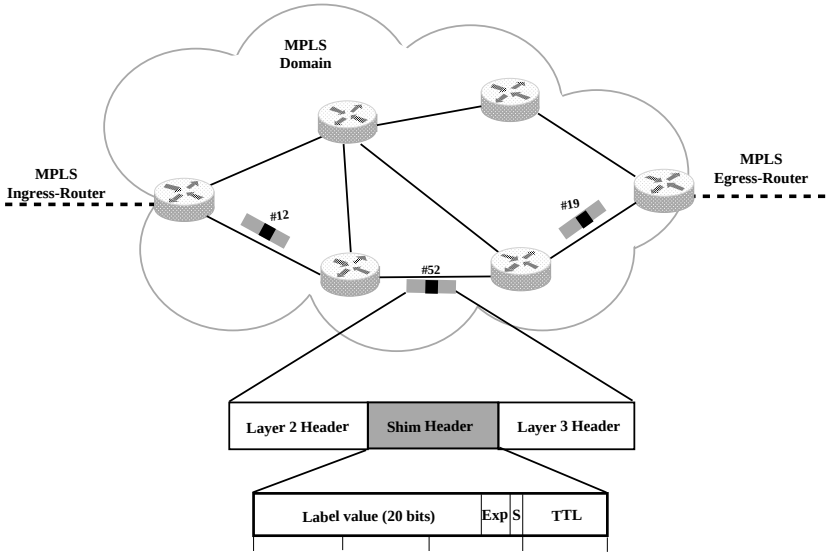

Figure 3.5: A network flow through an MPLS domain. The ingress router is responsible for adding a shim header which contains the label information. The interior—or core—router handles the traffic based on the MPLS label.

The MPLS forwarding component applies label switching forwarding tables that are maintained by the control component. These tables consist of a sequence of entries, each containing at least an incoming label (per input interface), an outgoing label, and an outgoing interface. Incoming packets carry a label, which is a 20 bit value that is typically encoded in an additional shim header of a multiple of 4 bytes between the link layer header and the network layer header. For switching techniques such as ATM, the label encoding is optimized by using the existing hooks on the data link layer. Based on the well defined label encoding the MPLS control component identifies incoming labels and usually replaces them by a different outgoing label before it actually forwards the packet to the relevant outgoing interface.

Figure 3.5 illustrates the basic concept of a flow traversing an MPLS-domain. It also lists the fields of the shim header [110]:

- An encoded label of 20 bits. Note that the identity of the network layer protocol is inferable from the value of the label.
- A three-bit field for experimental use. This experimental field (EXP-field) is often used to propagate IP-QoS related parameters to the control component of MPLS.
- A Bottom of Stack (S) bit which indicates whether the label is the last one of a label stack. By introducing this bit, MPLS allows the construction of hierarchical labels and thus the setup of nested MPLS domains.
- An eight-bit Time-To-Live (TTL) field which is used for loop prevention.

A drawback of the insertion of potentially multiple shim headers is the risk of fragmentation. Packets which were formally transmitted without any fragmentation may now exceed the Maximum Transmission Unit (MTU) of the underlying link. The MPLS label encoding document [110] specifies the reaction to this situation:

- Every ingress router of an MPLS domain should support a configuration parameter known as the ”Maximum Initially Labeled IP Datagram Size”. If an unlabeled IP datagram exceeds this threshold before labeling, the datagram must be broken into fragments, each of whose size is no greater than the value of the parameter, and each fragment must be labeled.
- If a labeled IP datagram is ”too big”, and the Do not Fragment (DF) bit is not set in its IP header, then the router may silently discard the datagram.
- If the router chooses not to discard a labeled IP datagram which is too big, or if the DF bit is set in that datagram, the router must leave the label switching mode and determine the DF bit value from the IP header. Whenever the DF bit is not set, the packet is fragmented. For packets where the DF bit is set, the packet is not forwarded and an ICMP Destination Unreachable Message is responded.

It is important to note that the main motivation for MPLS is not the improvement of the forwarding throughput, nor the provision of new QoS mechanisms. Using MPLS encoded labels avoids to switch to the IP-layer processing routines of the router and, if implemented in hardware, can therefore improve the throughput significantly. However, the fundamental design goal of MPLS is the ability to influence the operation of the control component. Traffic engineering (TE) is one of the major driving factors for MPLS.

Another benefit of MPLS is caused by its path-orientation. MPLS can potentially provide faster and more predictable protection and restoration capabilities in the face of topology changes than conventional hop-by-hop routed IP systems. Whenever a link fails, routing protocols are involved to recover the situation and to reestablish the routing convergence again. Unfortunately, the link-state protocols such as Open Shortest Path First (OSPF) do use timer mechanisms with exponential backoffs to reduce the frequency of link-state update propagation. Even in an excellently tuned system, the recovery time is of the order of seconds. For most layer one and two links, this time is increased by the lack of feedback mechanisms. Multiple “Hello”-messages are involved to identify a broken link, which again takes of the order of seconds. Finally, the shortest path calculation and the flooding of the new link state takes a while. MPLS offers three particular recovery mechanisms, all involving the use of a pre-configured, but unused backup tunnel:

- Link protection uses a backup tunnel which connects to the peered router of the protected link. In case of failures, the upstream router simply pushes an additional label to the MPLS label stack and transmits the packets to the backup tunnel. On the receiving router, so-called global labels are used to accept the arriving packets of different switched paths. The underlying idea is to forward a given label independently

from the incoming interface. Note that a single link protection is in principle capable of protecting multiple switched paths.

- Node protection follows a similar concept, but connects to the downstream router of the peered node.
- With path protection the backup tunnel covers major parts of the switched path using a diversely-routed path. Note that path protection does not necessarily connect the ingress router with the egress router.

The most efficient recovery approach is link protection. When applied to Packet Over SONET (POS) interfaces, which provide convenient feedback mechanisms for link failure detection, recovery can be kept within strong time constraints.

End-to-end connectivity is established by a consistent path of label pairs between ingress and egress router. Let $1, ..., n$ be the ordered list of routers in the network which have to be passed to transport data packets. As stated above, all routers maintain a list of labels per interface. Let I_j^i be a set which contains the incoming labels of router j on interface i and O_j^i be a set which contains the outgoing labels of router j on interface i.

We can establish an end-to-end connectivity based on label switching if for all $1 < j \leq n$, there exists some i_1, i_2, such that we find a label $L \in O_{j-1}^{i_1}$ and $L \in I_j^{i_2}$ under the constraint that interface i_1 of router $j-1$ is connected to interface i_2 of router j. As a result, we receive a uni-directional tunnel which connects the router 1 with router n. This tunnel is also called Label Switched Path (LSP). Figure 3.6 illustrates such an LSP traversing router $(R1, R2, R3)$. In this figure, the operation of control component relies on an Interior Gateway Protocol (IGP) which facilitates the full view of the topology. This knowledge is used by each router to locally bind a label for every possible destination address. The propagation of the local label binding between peered routers is done by a special Label Distribution Protocol (LDP) [6]. In this context, LDP is used to distribute the label binding information between peered routers. One possible end-to-end label distribution scenario is based on the downstream-on demand principle [111, 6]. The downstream router receives a label binding request from its peered upstream router and responds with its local label binding. Thus labels are ”downstream-assigned”, and label bindings are distributed in the ”downstream to upstream” direction. In the example illustrated in Figure 3.6 the label distribution starts with ingress router $R1$ which is requesting the local binding information from its downstream router. Hence, $R2$ receives the binding request from $R1$ and responds with its local label binding. $R1$ can now create a matching label entry for R2 on the relevant outgoing interface. Similarly, $R2$ receives the binding from $R3$. The notation “downstream-on-demand” originates from the fact that this process is initiated by the ingress router which is asking its peered downstream router to propagate its local label binding information.

As mentioned above, the control component usually consists of a routing protocol like Open Shortest Path First (OSPF). However, MPLS allows the integration of further con-

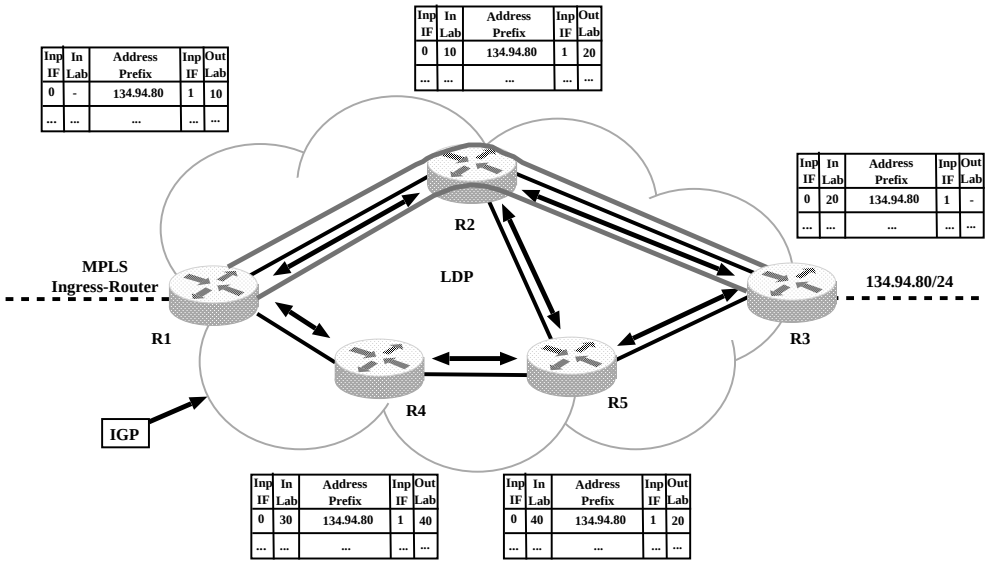

Figure 3.6: A Label Switched Path (LSP) shown as a tunnel in an MPLS domain. The LSP was created using downstream label binding distribution.

straint based routing algorithms. By decoupling the path of a packet from classical routing MPLS facilitates the setup of paths by a variety of ways, including constraints. In extending the RSVP signaling protocol [8, 7] by necessary objects, packet formats, and procedures required to establish and maintain explicit label switched paths (LSPs), the originator of the related RSVP PATH message obtains the full control about the mapping of an LSP. Similar extensions were proposed to LDP [36] including one Type-Length-Value (TLV) message field for explicit routing. The definition of an explicit route object facilitates the specification of the list of routers that comprise the most suitable path through the MPLS domain. This is analogous to IP source routing, where the instantiating router dictates the path through the network. Because the IP control plane is stateless, source routing has to be carried in all packets. Using the explicit route object to fix the path of an LSP, we can avoid this overhead. This capability provides a powerful mechanism for any bandwidth broker as it now can actually control the mapping of requests to the actual topology.

In addition to the explicit routing of LSPs for traffic engineering, the signaling mechanism for LSP set-up permits the specification of QoS attributes for the LSP. By assigning advanced capabilities to each link, bookkeeping functions of each router participate in the LSP signaling process by constraining the shortest path algorithm with additional constraints. The available bandwidth of a particular link is a typical example in this environment. In this scenario, bandwidth is not actually reserved by means of reservation capabilities of the router. Instead, it is used as a bookkeeping function for the available per-link capacity and it relies on an underlying congestion management configuration. This information is then used as an additional constraint in the standard shortest path calculation. Hence, LSPs are automatically mapped to the topology. However, there are two

deficiencies of this approach. First, constraints are not flexible. The available bandwidth is typically the only possible constraint. Second, this concept does still need additional mechanisms to actually instantiate the guarantees. A middleware based approach in which traffic engineering is performed by some middleware service allows the specification of more flexible constraints and is thus the better approach.

The Differentiated Services (DS) architecture is an appropriate technique to actually instantiate bandwidth guarantees. DS can be supported by MPLS in two ways. The 3-bit experimental (EXP) field in the MPLS shim header can be used to refer to a particular aggregate behavior. However, this limits the service differentiation to a subset of 8 of the possible 64 DSCP values. An LSP that marks the PHB in the experimental field is referred to as EXP-inferred-LSP (E-LSP). Core routers are then statically configured to implement an aggregate behavior per EXP-value. Whenever two LSPs with the same EXP-value pass the same output interface of a router, they share the assigned PHB. It is important to note that the EXP-field is only defined for the shim header of MPLS. As stated above, the shim header is not used for some native switching techniques, such as ATM. In this case, the use of E-LSPs is not possible without leaving the native switching mode, i.e. E-LSPs can only be transferred between end-points which do have their own IP address.

The aggregate treatment can alternatively be inferred by the label itself. An LSP is in this case referred to as label-only-inferred LSP (L-LSP). The configuration of the core routers for L-LSPs is dynamically updated by incorporating the packet scheduling rules into the LSP-signaling process. In some sense, L-LSPs can be compared to the IS model, in which request signaling is used to instantiate the related service for a particular flow. Here, however, signaling is not done on a per-flow base, but on a per-LSP base. L-LSPs use the experimental field of the MPLS shim header for other purposes such as the propagation of the drop precedence.

Comparable to the packet classification performed at the edge of a DS domain, the labels are assigned to incoming packets at the edge of the MPLS domain.

In addition to the pure mapping of flows or aggregates to LSPs, the edge router can also be used to police the traffic entering an LSP. Traffic policing allows to make assumptions about incoming traffic and is a fundamental building block for a formal analysis of the service behavior.

3.4 Network Calculus: Formal Aspects of Network QoS

The provision of network QoS prioritizes some packets higher than others. If all packets would be handled equally, no better-than best-effort services could be built. Service differentiation based on packet scheduling and traffic policing are fundamental building blocks for the provision of QoS. Traffic policing assures specific rules the incoming traffic follows.

Packet scheduling guarantees packet service times. Combining these assumptions, the service of each node can be formally modeled by a characterization of the outgoing traffic profile. By cascading this model to a system synthesized of a set of nodes, end-to-end behaviors can be derived.

Formalizing the policing function is a fundamental step for an analytical service model. The concept of an arrival curve as presented by [14, 26] facilitates the formal representation of policing functions and traffic profiles. Describing data flows by means of a cumulative function $R(t)$, defined as the number of bits seen on the flow in time interval $[0, t]$, the arrival curve is defined as follows:

Definition 6 *Given a wide-sense increasing function α defined for $t \geq 0$, we say that a flow R is constrained by α if and only if for all $s \leq t$:*

$$R(t) - R(s) \leq \alpha(t - s)$$

If R is constrained by α, α is called an arrival curve of R. R is also called to be α-smooth.

Because R's property $R(0) = 0$[1], an arrival curve gives an upper bound for the incoming traffic at any time t. It even gives an upper bound of traffic in a given interval $[s, t]$, i.e. it limits the amount of incoming traffic to $\alpha(t - s)$.

While $R(t)$ represents the cumulative arrival function, $R^*(t)$ denotes the cumulative departure function. Hence, we can describe the amount of bits that are held inside the system—the backlog—by $R^*(t) - R(t)$. Based on the assumption that any packet of a single flow is served in order of their arrival, we can also express the delay that would be experienced by a bit arriving at time t—called the virtual delay—as $d(t) = \inf\{T : T \geq 0 \, and \, R(t) \leq R^*(t + T)\}$.

The delay bound of the Guaranteed Service [117] is based on an arrival curve $\alpha(t) = min(M + pt, rt + b)$. Here, M specifies the maximum size of a datagram, p the the peak at which the source may inject bursts, r the rate the policing token bucket operates at, and b the token bucket depth. The related 4-tuple (p, M, r, B) is that what is called TSpec and what is used in the RSVP-PATH messages. Figure 3.7 shows this function.

Combining the concepts of arrival curves and token buckets allows to state that a flow policed by a token bucket (r, b) is constrained by an arrival curve $\gamma(T) = rT + b$.

Based on the concept of arrival curves, the handling of traffic in networking devices can be modeled by service curves [14, 26]. While the arrival curve formulates an upper bound for the traffic arriving at the forwarding engine of routers, the service curve defines a lower

[1]This property allows the construction of a valid arrival curve α^*, with $\alpha^*(0) = 0$ and $\alpha^*(t) = \alpha(t)$ for all $t > 0$.

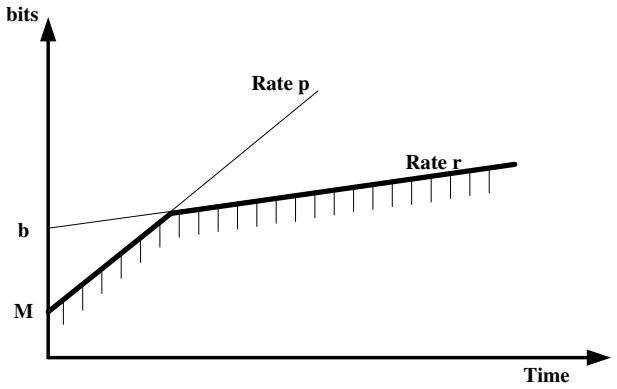

Figure 3.7: Arrival curve $\alpha(t)$ (thick line) for Integrated Services flows [117]. For any interval $[t_0, t_1]$ with $t_0 \geq 0$ the amount of traffic entering the system is limited by $\alpha(t_1 - t_0)$.

bound for the actual perceived departure function on the output link of the device. The definition is driven by the intuition that for any time t there exists some t_0, $t_0 \leq t$, at which the amount of bits leaving the node is greater or equal to the value of the service curve plus the backlog data at time t_0, i.e. the data which was already queued. Generalizing this abstraction to a general bit processing system S, i.e. a network topology consisting of multiple links and routers, the extended service curve [13] is introduced:

Definition 7 *Consider a system S and a flow through S with input and output function R and R^*. We say that S offers to the flow an extended service curve β if and only if for all $t \geq 0$, there exists some $t_0 \geq 0$, with $t_0 \leq t$, such that*

$$R^*(t) - R(t_0) \geq \beta(t - t_0)$$

In this dissertation we will omit the word "extended" and use "service curve" as synonym for the "extended service curve". The definition of the service curve means that for all $t \geq 0$, $R^*(t) \geq \inf_{s \leq t}(R(s) + \beta(t - s))$. We will abbreviate this relation by using the Min-Plus convolution operator $\otimes$ with $R^* \geq R \otimes \beta$. A service curve is called a strict service curve if and only if $R^*(t) - R(t_0) \geq \beta(t - t_0)$ for any $t_0 \leq t$.

A service curve of particular interest is the rate-latency function. The specific importance of this type of service curve is that it models the behavior of a non preemptive priority node, i.e. a node that is capable of serving a flow at a given constant, once a packet already in transit is served. Intuitively this behavior can be modeled as the concatenation of a node with a guaranteed rate r and a node with a maximum delay T.

Figure 3.8 shows this type of service curve. The Integrated Services model for a router is that the service curve offered to a flow is always a rate-latency function. As we will see, the expedited forwarding scheme of the Differentiated Services architecture can be

implemented using the same mechanisms in the router and can thus serve aggregates with a rate-latency service curve.

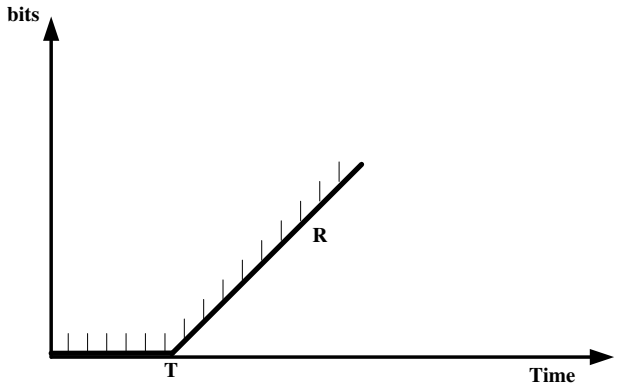

Figure 3.8: Rate-latency service curve: $\beta_{R,T}(t) = R[t-T]^+$. Here, $[t-T]^+$ is defined as $t-T$ for $t \geq T$ and 0 otherwise. The service curve guarantees that for any time t there exists some $0 \leq t_0 \leq t$ at which the amount of bits leaving the system is at least $\beta_{R,T}(t-t_0)$ plus all data backlogged in the system at time t_0.

The particular interest in rate-latency functions is caused by the fact that the packet schedulers of commodity routers are capable of providing this service behavior. An example is a non-preemptive priority node based on non-preemptive priority queuing and a FIFO layer 2 device queue. This scheduler uses two different queues. One queue is reserved for the priority traffic, the other one is used for the best-effort class. Whenever the scheduler is making a forwarding decision, it first serves any existing packet in the priority queue. However, as forwarding itself can only be done at link speed C, the actual forwarding of the high priority packet might be delayed by a low-priority packet of size l^L_{max} which is currently in transit, i.e. which is currently located in the device queue. If we call R^*_H the output rate of the high-priority flow and fix some time t and call s the beginning of the backlog period of the high-priority flow, we receive:

$$R^*_H(t) - R^*_H(s) \geq C(t-s) - l^L_{max}$$

and thus:

$$R^*_H(t) \geq R^*_H(s) + [C(t-s) - l^L_{max}]^+$$

Because of the fact that s is the beginning of the backlog period, the backlog of high priority packets is 0 and thus $R^*_H(t) = R_H(t)$. We now do have a rate-latency function with rate C and latency $\frac{l^L_{max}}{C}$.

In addition to non-preemptive Priority Queuing (PQ), Weighted Fair Queuing (WFQ) is a popular scheduling discipline for the provision of a rate-latency service curve too. The concept of WFQ is quite similar to that of PQ. However, it allows the differentiation between more than two service classes, and it limits the rate at which each class is served. Some hardware vendors implemented the latter feature for priority queuing as well, as the

ability to limit the available amount of priority bandwidth somehow offers an additional way to prevent starvation of the best-effort traffic. If g is the assigned rate to the particular flow, we receive a rate-latency function of rate g and latency $\frac{l^H_{max}}{g} + \frac{l^L_{max}}{C}$.

Using the formalization of arrival and service curves, we can calculate worst-case boundaries for the delay and the backlog. Figure 3.9 illustrates the calculation for a system which is offering a rate-latency service curve to a token bucket constrained flow. In general, the following three important boundaries can be derived [14]:

Theorem 1 Backlog Bound: *Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve β. The backlog $R(t) - R^*(t)$ for all t satisfies:*

$$R(t) - R^*(t) \leq \sup_{s \geq 0}\{\alpha(s) - \beta(s)\}$$

Given any bit processing system S which is offering a service curve of β, this formula enables us to calculate the backlog for any arrival curve at any time. Similarly, we can calculate the delay bound by the following formula:

Theorem 2 Delay Bound: *Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve β. The virtual delay $d(t) = \inf\{T : T \geq 0 \, and \, R(t) \leq R^*(t+T)\}$ for all t satisfies:*

$$d(t) \leq \sup_{s \geq 0}(\inf\{T : T \geq 0 \, and \, \alpha(s) \leq \beta(s+T)\})$$

The term $h(\alpha, \beta) = \sup_{s \geq 0}(\inf\{T : T \geq 0 \, and \, \alpha(s) \leq \beta(s+T)\})$ is also called the horizontal deviation h.

Theorem 3 Output Flow: *Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve β. The output flow is constrained by the arrival curve:*

$$\alpha^*(t) = \sup_{u \geq 0} \alpha(t+u) - \beta(u)$$

The term $\sup_{u \geq 0}\{\alpha(t+u) - \beta(u)\}$ is also written as $(\alpha \oslash \beta)(t)$. In this context, $\oslash$ denotes the Minus Operator of the Min-Plus-Algebra [14]. Similarly, a service curve can be defined using the Min-Plus Convolution:

$$(f \otimes g) = \inf_{0 \leq s \leq t}(f(s) + g(t-s))$$

A system offers a service curve β if and only if $R^* \geq R \otimes \beta$. The Min-Plus Convolution allows an efficient formalization of the end-to-end behavior of a flow. A flow that is passing

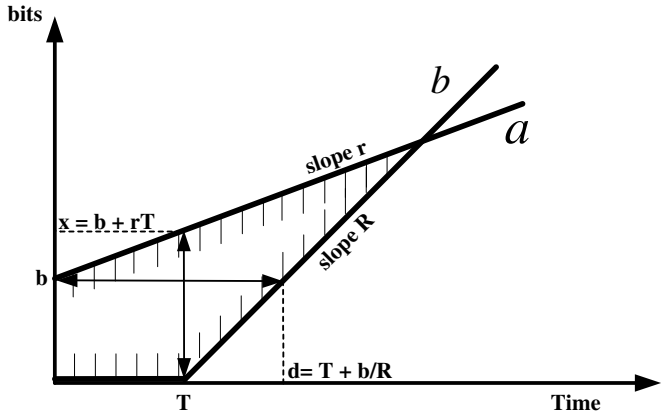

Figure 3.9: Buffer bound $x = b + rT$ and delay bound $d = T + \frac{b}{R}$ for a System S which arrival curve is constrained by a token bucket (r, b) and which is offering a rate-latency service curve $\beta_{R,T}$. Note that we assume that $r \leq R$.

a system consisting of a sequence of routers $i = 1, ..., I$, each with a service curve β_i is served with a service curve of $\beta = \beta_1 \otimes ... \otimes \beta_I$. This behavior can easily be derived from the definition of the service curve and the associativity and the isotonicity of the min-plus convolution. Hence, it can be established that the worst case delay over a concatenation of nodes is less than the sum of the worst case delay at every router. This specific property is also called the pay-bursts only once principle.

3.5 Conclusion

This section presented an overview about two architectures supporting IP-based network guarantees: The Integrated Services (IS) architecture and the Differentiated Services (DS) architecture. While the IS framework provides service guarantees based on a flow-based packet differentiation, the DS architecture differentiates the treatment of aggregates. By specifying per-hop behaviors for classes of aggregates it facilitates the provision of service guarantees.

Though the IS is able to provide strong QoS guarantees, its practical use suffers from significant scalability problems. The DS framework addresses this issue by its concept of aggregation. The price for building aggregates is, however, a weaker control over the provisioned service parameters. In this context, traffic engineering capabilities offer the opportunity to control the compound of an aggregate and can thus be used to provide stronger service guarantees. The MultiProtocol Label Switching (MPLS) architecture is a good candidate to extend the ability to build reasonable end-to-end services based on the DS framework.

The benefit of traffic engineering can be expressed more formally. This chapter described the fundamental concept of the network calculus, a theory of deterministic queuing systems found in computer networks, which is a potential vehicle to express delay boundaries for a particular DS environment.

An important aspect for building network services based on aggregates is the dynamic assignment of flows to aggregates. A specific management entity, called a bandwidth broker, is introduced to handle this dynamic mapping for a single trust domain. From a Grid perspective, bandwidth brokers can be viewed as a local resource management system for the network services. It is therefore clear that the integration of a bandwidth broker into a general resource management framework of computational Grids is a natural extension. The straight forward approach of providing network QoS based on a managing entity controlling the access to aggregates makes this concept a good candidate technology for the provision of network guarantees in computational Grids.

Chapter 4

Requirements for a Grid Bandwidth Broker

The specific network requirements of Grid applications are addressed by the introduction of a particular middleware service: the bandwidth broker. The concept of a bandwidth broker is typically bound to the Differentiated Services (DS) architecture. Its fundamental task is to facilitate and control the dynamic access to network services of a network domain. By carefully limiting the traffic admitted to the traffic aggregates, QoS guarantees for bandwidth and further metrics can be provided. An important issue for the provision of end-to-end guarantees is the availability of QoS-supporting mechanisms in all domains the traffic traverses. In that context, bandwidth brokers can also be used to facilitate a consistent end-to-end service model, even if some transient domains do not provide services on top of DS, but are capable of providing the requested capabilities by some other mechanism, such as over-provisioning. From the perspective of a Grid resource management, a bandwidth broker intermediates between the Grid resource management and the actual underlying capabilities of the network. Hence, bandwidth brokers are a vehicle for encapsulating the details of the underlying networking from the Grid resource management. This allows the integration of a variety of different QoS techniques into the Grid resource management. Figure 4.1 illustrates this scenario.

4.1 Integration into Computational Grids

In a Grid environment access to services is often facilitated by the use of specific local resource managers, which control the access to the resource specific services. A typical example is a batch sub-system installed on a particular supercomputer which is the interface to the underlying computing resource. The Grid middleware uses these local resource

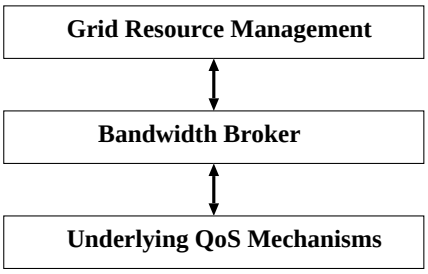

Figure 4.1: The bandwidth broker introduces an additional middleware layer between networking capabilities and the Grid resource management. Though bandwidth brokers are typically associated with the Differentiated Services architecture, they can, in principle, also be used in different QoS scenarios to signal the availability of a particular guaranteed network capability to a requester.

managers to build a homogeneous service infrastructure. From the user perspective, Grid resources should be accessible by a common Application Programming Interface (API). From a Grid resource management perspective, services should be divided into more general parts which are common for all service operations in this virtual organization, and those parts which are specific to this particular resource manager. While the request authentication, its basic authorization, and the event notification are common for all service requests in a particular Grid, the admission control depends on the local resource manager. When network services are offered by bandwidth brokers which are in this context acting as a local resource manager, the Grid resource management system has to interface them.

4.1.1 Grid Resource Management

A Grid resource management service translates a given service request to an actual allocation of the request on a set of resources. While the research and development of a Grid middleware is an ongoing effort for several years, there is no standardized resource management service available yet. However, there are widely accepted frameworks available. A good candidate is the Globus Resource Allocation Manager (GRAM) [27] of the Globus Toolkit [50]. Because the management focus of this architecture is on the allocation of CPU resources, the integration of a fundamental new resource type should not necessarily be part of this architecture, but should extend its capabilities in a compatible way.

4.1.2 Coordinated Reservation and Allocation of Resources

Specialized resources required by high-end applications such as high-bandwidth virtual channels, low-delay steering channels, or scientific instruments and supercomputers are in scarce and in high demand; in the absence of advance reservation mechanisms, coordination of the necessary resources is difficult. The success of claiming such a service,

especially when an application is trying to combine these, would rely on the current availability without any deterministic possibilities to schedule the use in advance.

Additionally, several Grid resources are not time-sharable. Consider a single virtual reality device or a wind tunnel. Those devices are typically used in an mutually exclusive operation mode and are often pre-assigned, i.e. a particular wind tunnel is reserved. In a Grid environment, those resources might be scheduled together with computing capabilities and the network in between. To ensure an economic use of each resource, reservation mechanisms are desirable to ensure that resources and services may be scheduled in advance.

Because bandwidth brokers are part of a Grid resource management system, they should support this request scenario, i.e. they must accept resource capability requests with a future starting time. In this context, the typical immediate reservation is a subset of the more general advance reservations request with the current time as start time.

Another important issue of advance reservation for network capabilities is the ability to accept a service request without the full knowledge of the endpoints, i.e. a service request between two IP end-points without actually knowing the port numbers of both end-points. This scenario is quite typical in a Grid environment were the resources are known in advance, but the application has not yet started and dynamic ports have not yet been selected. To instantiate such a reservation, a bandwidth broker must be able to refine a given request during its lifetime. This capability is also important in the context of aggregated reservations. An aggregated reservation can be claimed in chunks by multiple authorized subjects.

4.1.3 Application Interface

Access to services should be provided by multiple layered interfaces each of it providing the necessary level of abstraction. An example for a high-level interface is a seamless graphical user interface, while a medium-level interface could be an API which allows the coordinated use of multiple different resources. A typical low-level interface is the API which interacts with the Grid resource management service. This API is built on top of other Grid services such as security and notification.

4.1.4 Policy Information for Distributed Authorization

Emerging computational Grid environments promise new capabilities for problem solving and effective distance collaboration [52]. However, applications that use Grid technologies often place substantial demands on scarce—and typically shared—resources such as networks, storage systems, and computers. Because these resources are both scarce and shared, a system of rules for resource use, or *policy*, is often associated with a resource

to regulate its use [131]. *End-to-end* performance guarantees typically require the *co-reservation* of multiple distinct resources. A number of technical issues complicate the co-reservation process:

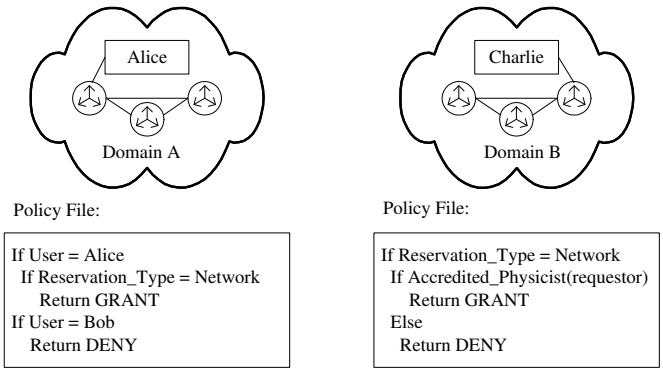

Figure 4.2: Different domains may have different reservation policies. Even the policy language might be different.

- *Policy heterogeneity.* Co-reservation can require negotiation with resource owners in each of several distinct administrative domains. Each domain may have different policies governing who can use its resources and for what purposes, and different trust relationships with individual users. For example, in Figure 4.2, domain A's policy might state that "Alice can use the network, Bob cannot," while domain B's policy is that "only accredited physicists can use the network."

- *Trust heterogeneity.* Scalability demands that every resource should not have a direct trust relationship with every user. While some domains know about individuals (e.g., domain A), others must be able to delegate responsibility for personal trust relationships to third parties. (For example, domain B agrees to provide resources to anyone whom a third party accredits as a "physicist.")

- *Interdomain policy dependencies.* A policy expressed in one domain can be dependent on policy decisions expressed in other domains. For example, for reasons presented below, domain A may wish to enforce the policy "I will only authorize a reservation if reservations have also been approved for all other resources in the end-to-end path." Or, domain B might only authorize bandwidth greater than 10 Mb/s if domain A has committed to shaping the traffic in a certain way.

- *Scalability.* If a set of applications creates many parallel flows between the same two end-domains, it is infeasible to negotiate an end-to-end reservation for each one.

4.1.5 Deployability

Grids are built by user communities to offer an infrastructure helping the members to solve their specific problems. Hence, the geographical topology of the Grid depends on the distribution of the community members. Though there might be a strong relation between the entities building the virtual organization, a Grid still consists of resources owned by different, typically independent organizations. Heterogeneity of resources and policies is a fundamental result of this. Grid services should therefore follow a pragmatic deployment concept, i.e. they should avoid to rely on to specialized assumptions such as specific rare operating system capabilities. We will call this the “easy-to-deploy” paradigm.

The focus on commodity hardware products and operating systems is an important aspect for the applicability of any Grid service. Solutions which are built assumptions of specific new hardware or protocol capabilities should be avoided. With respect to a bandwidth broker this paradigm has the consequence that the network services should not rely on the assumption of a complex set of aggregates. The more complex the assumptions are, the more difficult it will be to establish effective end-to-end guarantees in a multi domain environment. Hence, the focus should rely on services which are already under consideration in major research network providers. This is, of course, the most demanding service: the Premium Service which is built on top of the EF aggregate.

4.2 Traffic Engineering

The fundamental task of a bandwidth broker is to facilitate and control the access to the better-than best-effort network services of a particular domain. Each service request enters and leaves the domain at a specific node, regardless whether the end-points are located in a single domain or in different ones. Of course, these requests have to pass the admission control procedure implemented by the bandwidth broker. In its simplest way, this procedure neglects the actual end-points of the service request. This solution limits the offered service to the achievable service of the link with the minimum QoS capability of the domain, i.e. one must assume that all requests will flow through this particular link.

A more advanced admission control procedure would use the knowledge about the network topology and about the routing tables, and would identify the actual path of the request in the controlled domain. In that case, the service would not be limited by the minimum link capability anymore. However, also this approach does have its limitations. Service requests might exceed the assigned share of the QoS capabilities of a single link, because flows were mapped to the topology based on standard IP routing mechanisms though there are alternative paths which could serve the request. Hence, if the bandwidth broker would be able to actually select a different path in the network topology of the controlled domain and to enforce this path for the request, more service requests could be served. Figure 4.3

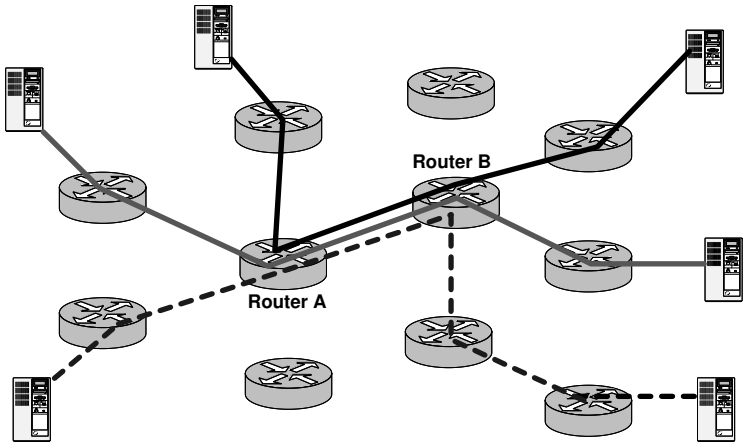

Figure 4.3: The mapping of three service requests onto the topology of a network domain. The figure omits the illustration of the interconnecting links for readability reasons. All three reservations pass through a common link between router A and router B which might become a bottleneck.

illustrates a scenario where three requests pass a common link in the topology. This potentially creates a capacity bottleneck or might cause delay variance problems. The solution illustrated in Figure 4.4 uses traffic engineering to place the reservation efficiently on the existing topology.

While the increase of acceptable service requests is a win by itself, the importance of traffic engineering capabilities is increased by the demand of the provision of advanced service parameters such as delay and jitter boundaries in the fuzzy context of aggregate based scheduling. Using standard Differentiated Services features service parameters are influenced by the constitution of the aggregate, i.e. by the amount of flows in a given aggregate, and by the topology, i.e. by the amount of multiplexing points where packets of the same aggregate traverse from different input links to the same output link [21]. In absence of the ability to influence the mapping decision, a worst case calculation for both, topology and packet arrival, is the only strict bound a bandwidth broker can provide.

In a Grid environment the demand for traffic engineering capabilities is even stronger as those features could facilitate

- the isolation of specific requests for the support of large-scale bulk transfers with deadline support.
- the satisfaction of heterogeneous services demands with a minimum of required aggregate behaviors, i.e. with the “easy-to-deploy” paradigm.

Of course, a link based admission control and the active influence of paths do have their drawbacks. First, it is unlikely that each single micro-flow will be explicitly mapped to the topology. The associated overhead would often exceed the benefit. This dissertation therefore proposes to use this technique in transient domains, where service requests are

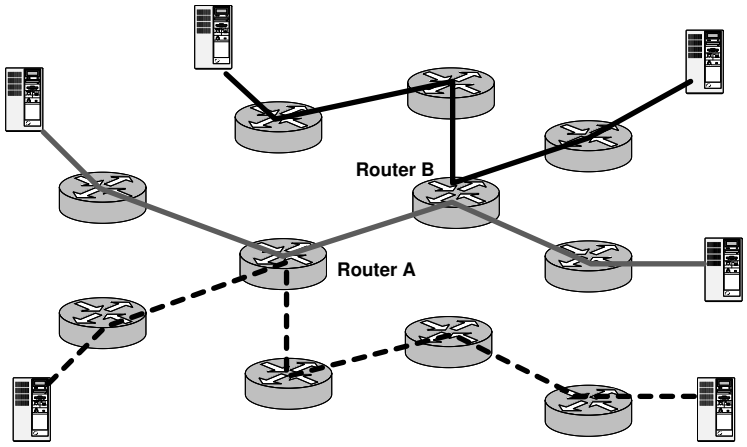

Figure 4.4: The mapping of three service request onto the topology of a network domain using traffic engineering mechanisms. The applied traffic engineering algorithm avoided the creation of a bottleneck link between router A and router B as shown in Figure 4.3. The figure omits the illustration of the interconnecting links for readability reasons.

typically aggregated. Second, whenever flows are mapped to the underlying topology, either by using IP's shortest path calculation or by adding more convenient constraints, the problem of consistency arises. IP-robustness is perceived by its stateless property. A bandwidth broker therefore has to assure that it maintains a consistent view of the topology and the actual paths of the requests.

4.3 Advanced Networking Demands

In a Grid environment the provision of QoS mechanisms has to address a complex mixture of flows, from low bandwidth to high bandwidth and from low latency to high latency. In addition, flows may change their requirements dynamically throughout their lifetime which is an important issue. Section 2.3 identified two basic services which should be provided by a bandwidth broker: a Guaranteed Rate service and a Premium service. This section relates these services with the protocol properties of the applications.

4.3.1 The Transmission Control Protocol

Establishing service guarantees for TCP flows is a challenge. This subsection reviews the TCP mechanisms that affect Quality of Service.

Flow Control

When TCP was first developed, it only used one mechanism to regulate the speed at which data could be sent. This mechanism, called flow control [122, 25], ensures that TCP does not send data faster than the receiver can accept the data. To do this, the receiver provides the sender with an *advertised window*, which is essentially the amount of free buffer space it has to accept data. From this advertised window, the sender calculates how much data it can actually send. This amount may be less than the advertised window if the sender has sent some data that has not yet been acknowledged by the receiver, since this data will fill the receiver's free buffer space once it arrives.

Ideally, an application that makes a reservation for network QoS makes sure that it does not send data faster than the reservation allows. However, TCP's flow control mechanisms cannot be controlled by the application, so it can be hard for the application to ensure that it sends data at a steady pace, even if it provides data to TCP at a steady pace. In fact, TCP may send data in short high-speed bursts that may be faster than a reservation allows. For example, the sending application may provide data steadily to TCP, but the receiving application may be too busy to consume the data, so the receiver provides a very small advertised window. When the receiver is suddenly able to consume the data, the advertised window may become very large, which enables the sender's TCP to send the carefully paced data in one large burst.

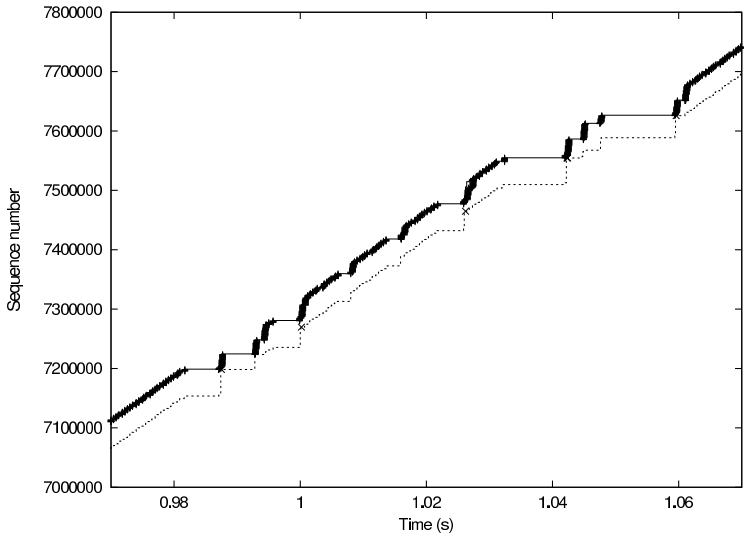

Figure 4.5: Demonstration of TCP's flow control. The graph illustrates TCP's sequence numbers over time within the boundaries of TCP's sliding window. The slope indicates the throughput. The traced flow, given by the thick black line, is driven by the evolution of the sliding window and reacts to state changes of the network. This effect is also known as TCP's self-clocking feature.

Figure 2.3 displays a network trace of an application which is writing with 128 kilobyte (KB) to the socket buffer. The advertised window was significantly larger than the application's writing speed, means that TCP was able to write each chunk of 128 KB at link speed. This type of application is not limited by TCP's flow control mechanisms, it is limited by the application which is not able to provide data fast enough.

For many types of applications, the actual throughput is limited by TCP's flow control mechanism. Figure 4.5 illustrates such an example. Here, the output of the illustrated TCP flow is directly influenced by the sporadic background load. The resulting traffic profile of the flow varies from smooth to bursty. Any prediction of the actual short term behavior in these scenario is hard, since the applicability of the prediction strongly depends on the accurate estimation of the future networking conditions.

Other problems with assuring a guaranteed transmission rate for TCP are caused by advanced mechanisms of the implementation of the protocol stack. The use of delayed acknowledgments is supposed to reduce the number of acknowledgment packets which are not piggy-backed with some data. Because a socket buffer limited flows reacts to the arrival of acknowledgments, its actual traffic profile varies with the implementation of delayed acknowledgments, i.e. with the amount of packets that must be receipt before an acknowledgment is sent. The accuracy of a prediction therefore depends on the knowledge of the end-system parameters.

Congestion Control

In the early 1980's, it was realized that TCP's flow control was unfriendly when there was a lot of congestion in the network. For example, if the advertised window was large, but a number of packets were dropped at the router due to congestion, TCP would resent the data in one large burst. Since all senders on the network were behaving similarly, TCP's behavior encouraged the congestion, instead of avoiding it.

To react to congestion in a friendly manner, TCP essentially adopted three rules:

1. If a packet is lost, assume there is congestion.
2. If a small amount of congestion is detected (indicated by the receiver sending a duplicate ACK when packets are received out of order), decrease the window size by a small amount. Specifically, we divide the window size in two, then gradually increment it.
3. If a large amount of congestion is detected (indicated when we receive no ACKs, suggesting many packets were dropped), decrease the window size to one or a some other small value, then gradually increase it again.

```
Initialize W = 1 (or 2) and T with max_cwnd

(1) After every non-repeated ACK:
    if W < T, set W = W + 1; Slow Start Phase
    else set W = W + 1/[W]. Congestion Avoidance Phase

(2) When the number of repeated ACKs exceeds a threshold
    (fast retransmission/recovery) retransmit "next expected" packet;
    set T = W/2;
    set W = T; (i.e. halve the window)
    resume congestion avoidance using the new window

(3) Upon time expire, the algorithm goes into slow start:
    set T = W/2;
    set W = 1.
```

Figure 4.6: Description of Additive-Increase/Multiplicative Decrease (AIMD) algorithm for the window size evolution of a TCP Reno implementation (see [84] for details). Here, W represents the congestion window size and T is a threshold used to switch between the different phases, often referred to as *ssthresh*.

These rules are implemented by creating a *congestion window* [76, 84]. When deciding how much data can be sent, TCP uses the minimum of the congestion window and the advertised window.

Rule two is known as fast retransmit. Rule three is known as slow start [121, 3]. Slow start is also used when a TCP connection is first created. Slow start increases the congestion window exponentially when a connection is first created. If congestion causes slow start, the congestion window is increased exponentially until it reaches one-half its previous size, at which point it is increased linearly.

In short, when TCP packets are dropped, TCP slows down the rate at which it sends. For best-effort traffic, this is an excellent mechanism that helps prevent congestion collapse on the Internet. For an economic use of a GR service this is, however, a potential problem. When the advertised window is large and TCP sends a large burst of traffic, several packets in a row may be dropped, causing TCP to go into slow start mode and therefore to decrease its sending rate dramatically. The problem is that TCP interprets lost packets as an indicating of congestion, but here the dropped packets do not indicate congestion, but the exceed of the reservation. Therefore, it is in our best interests to prevent TCP from sending data in large bursts in order to avoid slowing down. As we will see below, this can be handled by using traffic shaping at the ingress router with the intention to avoid bursts.

The process of the congestion control mechanism is described more formally in [84]. Figure 4.6 gives a formal overview of the window evolution.

To emphasize the effect of these mechanisms, we note that the achieved transmission rate, also called goodput, of a TCP application is limited to the actual window size divided by the Round-Trip Time (RTT), i.e. by the bandwidth-delay product [122]. The impact of dropped packets on the short-term TCP goodput can be enormous. Multiple losses of packets within a single round-trip cause a malfunction of the fast retransmission algorithm of the most widely deployed version of TCP: TCP Reno [69]. According to the measurement

of Paxson [101], 13% of the observed TCP Reno traces contain a fast retransmit followed by a timeout. This situation changes TCP's state to slow start resulting in a significant impact on the achieved throughput. The Selective Acknowledgment (SACK) option [89] was introduced to improve TCP's behavior in response to multiple packet losses. SACK treats multiple losses within one round-trip as a single congestion signal and can therefore recover roughly within a single RTT interval. To timeout, TCP-SACK must loose a string of ACKs, or loose a retransmitted packet. While TCP-SACK increases the goodput of a connection in case of packet loss, it is still sensitive to drops.

First, the use of SACK together with the in high-bandwidth environments commonly used time-stamp-option limits the number of non-contiguous losses to three. Additionally, the congestion avoidance algorithms used in combination with SACK are often fairly conservative, which means that TCP is falling into congestion control phase, once a packet drop is recognized. This impacts the achieved goodput reasonably.

The macroscopic behavior of these mechanisms is described in [91, 84, 99]. Experimentation and simulation with TCP-SACK is published in [17, 38, 63]. The general impact of the window size on the goodput is evaluated in [98, 90]. Statistical analysis for the impact on a DS implementation is discussed in [114, 141].

A major shortcoming of these models is that they rely on specific assumptions which are not necessarily fulfilled in the heterogeneous Grid environment. Bolliger [12] presented in his dissertation a comprehensive overview about the listed throughput models. Of course, their general applicability relies on the accuracy of assumed network characteristics such as round-trip time and packet loss rate. It is widely agreed that modeling retransmission timeouts is problematic. Firoiu et. al. [45] state that most models do not work accurately for packet loss probabilities above 0.02. Even by modeling the effect of timeouts more explicitly [99], Bolliger [12] experienced side effects of the TCP variants and configuration parameters of the end-systems which limit the accuracy of the models as well as bandwidth fluctuations. He found that approximately 1% of the Reno connections experienced a throughput change between the first half and second half of their connection of more than a factor of 4 (up to a factor of 100!). Due to the heterogeneity of the Grid, the bandwidth models are useful to support a service level estimation, but are not designed to be a basic building block for a GR service or the support of deadline file staging.

4.3.2 The Berkeley Socket Interface

A widely deployed interface to implementations of the TCP protocol stack is provided by the Berkeley socket interface which was developed at the University of California at Berkeley as part of their BSD 4.1c UNIX version. The fundamental abstraction of this API is the representation of communication end-points as generic data structures called sockets [135]. The interface specification lists a set of operations on sockets in a way

that communication can be implemented using standard input/output library calls. It is important to note that the abstraction provided by sockets is a multi-protocol abstraction of communication end-points. The same data structure is used with Unix services as files, pipes and FIFOs as well as with UDP or TCP end-points.

Though the concept of sockets is close to that of file descriptors, there are, however, essential differences between a file descriptor and a socket reference. While a file descriptor is bound to a file during the open() system call, a socket can exist without being bound to a remote endpoint. Thus, binding sockets is a separate and important process with sockets. We will use a similar concept for all resource reservations in Section 5.2. The operations for creating and binding sockets are defined as follows:

```
socket(family, type, protocol)
bind(socket, local_address, address_length)
```

There are three types of socket interfaces defined with TCP sockets. For the set up of a TCP connection the function sequence is different for sender and receiver. While the sender issues the

```
connect(socket, destination_address, address_length)
```

call to perform the three-way handshake of TCP, the receiver has to issue two calls:

```
listen(socket, queue_length)
accept(socket, address, length)
```

An important aspect is the relation between the above listed call-sequence and the protocol processing of the TCP handshake. While the listen()-call is an asynchronous operation which is related to the receipt of TCP-SYN-messages, connect() and accept() are typically blocking operations. Figure 4.7 gives an overview about this relation. As a consequence, the execution time of a connect()-call can be used as approximation for the RTT at the current state of the network.

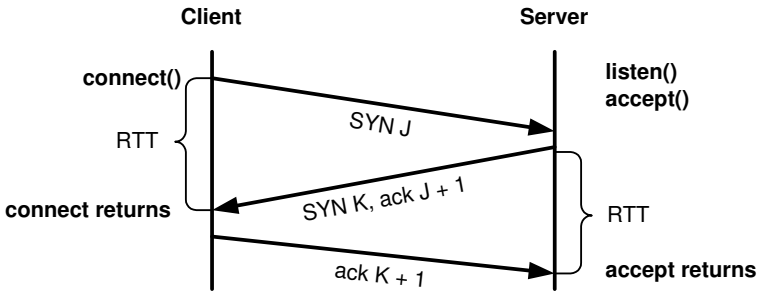

Figure 4.7: Functional processing of the three-way handshake. The execution time of the connect()-call is an estimate of the current round-trip time of the network.

Another aspect of the Berkeley socket interface is its relation to the achievable network throughput. Ignoring packet loss and congestion, the throughput of a TCP connection is fully determined by the socket-use profile of the application and the socket-buffer sizes.

The first throughput relevant parameter is strongly application dependent. Whenever TCP throughput is analyzed, most models assume a steady state in which applications can always provide data to the socket buffer. This is, however, not true for all type of applications. An example here is an MPI-based distributed supercomputer application which has a long term TCP connection between nodes, but does perform calculations of different complexity before it reads and writes to the socket buffer. Figure 2.3 demonstrated the impact of this application behavior. The frequency at which data is copied to the socket buffer causes TCP to have a fully opened window. Hence, the protocol stack injects the data at link speed. It is important to note that the ability to produce or consume data does also depend on competition for CPU cycles or disk access on the end-systems. A busy receiver or sender might produce a bursty TCP traffic profile even if the application itself is designed to operate in steady state.

The second throughput relevant parameter is directly influenced by the socket API. The underlying data structures of sockets are controlled and manipulated by a generic library call: the setsockopt()-call. With respect to the window sizes used by TCP, the attributes "SO_SNDBUF" and "SO_RCVBUF" control the maximum window sizes of the receiver and transmitter. Here, however, three constraints have to be noted:

1. The usage of any buffer size above 64 KB requires the TCP window scale option to be in place. This option, however, is negotiated during the three-way handshake of TCP. It is thus not possible to use arbitrary buffer sizes in the setsockopt()-calls at any time. Whenever the connect()-call or the listen()- respectively accept()-call was issued, the maximum window size is not changeable anymore.

2. Most operating systems use a system wide default parameter for the socket send and receive buffer. Additionally, they often define a system wide maximum value which cannot be exceeded, even though the related setsockopt()-call is using a larger value. A consequence is that the socket buffer limitations depend on the particular system configuration. Application developers do often not have the knowledge of the configuration parameters.

3. Some operating systems also shrink the maximum allowed congestion window size [130]. As a result of this, the maximum accepted advertised window might even be smaller than the socket buffer size, even if there is no packet loss at all.

This argumentation leads to the conclusion that any buffer tuning algorithm is limited by the lack of influencing the window-scale option directly. Oversizing the socket buffer, however, might increase the burstiness and is thus an option which requires a careful trade-off between risk and benefit, especially when this is applied to a policed GR service.

4.3.3 Bulk Transfers with Deadline Support

Ambitious applications with demanding TCP throughput requirements such as bulk-data transfers with a guaranteed deadline should rely on a Guaranteed Rate (GR) service, where no packets of a reservation conforming flow should be dropped. Otherwise TCP might leave its steady state and would go either into slow-start or congestion control phase, which would significantly reduce the achieved goodput.

In combination with the possible burstiness of several applications, any QoS framework supporting guaranteed deadlines for TCP-based file transfers should either support reasonable short-term bursts without dropping the packets explicitly, or must build mechanisms which avoid these bursts. The term “reasonable” depends on the desired peak rate and the expected round-trip time in the network, i.e on the required window size of the transmitter.

As mentioned above, the achieved goodput of an application depends on the following major factors:

- The rate at which the application is writing to resp. reading from the socket buffer,
- and the socket buffer size of receiver and sender with respect to the round-trip-time of the network in between.
- The amount and the distribution of dropped packets. Whenever a packet gets dropped, the TCP source reacts on this either by falling into congestion control mode, which halfs the current transmission rate, or, in case of a more complex loss scenario, by falling into slow start mode, which cuts of the goodput even more dramatically and requires a timer timeout in the order of several tens of a second.

Network applications have to balance those factors efficiently. Traditionally, they have to estimate their socket buffer write rate. This write rate can be viewed as the rate the network should at least be able to serve the related flow. However, it might be hard to estimate this as it depends on more then the application itself. The execution environment such as disk I/O and CPU competition has an influence on the actual rate. In the context of a Grid resource management, we can address this issue by offering the ability to co-reserve the end-host CPUs. Second, the application has to select an appropriate socket buffer size for a given connection. The important issue here is, as stated above, that the socket buffer size must be set before the connection is established. Unfortunately, it might be counterproductive to simply overestimate the socket buffer size in a best-effort environment. When congestion occurs sporadically, a large socket buffer introduces a more bursty traffic pattern. The ability to access a GR service which is capable of handling a full socket buffer size without any packet loss is a potential solution for these issues.

4.3.4 Control and Adaptation

The ability to control the actual use of the network and to react on state changes is essential in Grid environments. The support of bulk transfers with deadline support introduces additional control features which should be supported by a bandwidth broker. This is especially true if the specific need of such a transfer is addressed. It is designated to use unused service resources as long as no one else is actively requesting them. An automation which adapts the actual transmission rate to the state of the network and which still guarantees to meet the deadline is therefore a desirable feature. The ability to control and to adapt, also based on the existence of state feedback mechanisms, is a logical result of this desire.

In Grid environments, adaption is a common behavior of many applications. A teleimmersion application reacts to user actions by adapting the virtual world. For example, sensors detect the user's location and orientation to enable the simulation program to perform calculations to update what the user sees. This creates a highly dynamic environment, because the result of the simulation process is not known in advance. For example, one result of this calculation could be that a new object has to be illustrated and stored in the VR-database. Another example would be a major update of the visualized virtual world. Any portion of this process may be distributed: the raw data may be sent to the client's computer to be processed and visualized or processed data may be sent to the client's computer to be rendered [29, 49]. The related bandwidth requirements are difficult to predict. A QoS framework should therefore provide feedback information to the application which indicates when it has made a reservation that is too small or too large.

4.4 Conclusion

Bandwidth brokers facilitate and control the access to network services to their domain by encapsulating the underlying Quality of Service mechanisms from their users and can thus be viewed as the manager of the network as a Grid resource. In a Grid environment, users typically do not contact local resource managers directly. Instead, they use a common resource management framework that securely intermediates between the Grid and the resources. This leads to the conclusion that bandwidth brokers should be a part of this framework.

The specific service demand of Grid applications was already discussed in Section 2.3. This chapter extended the capability requirements for bandwidth brokers by the particular constraints of the complex Grid infrastructure. First, the demand for advanced network services is always associated with the demand for additional resources, i.e. the demand for capabilities which act as a data source and sink. The support for a coordinated reservation and allocation of multiple resources is thus a fundamental requirement for a Grid-enabled bandwidth broker. Advance reservation capabilities are an efficient vehicle to support this

task and are therefore a design requirement. Second, bandwidth brokers must be integrated into the distributed authorization environment of the Grid with complex policy dependencies. In this context, the network is a particular resource as it is typically organized by bilateral service agreements of administrative independent domains. Finally, bandwidth brokers, when applied in a Differentiated Services (DS) environment, should be capable of offering the required services for different packet differentiation rules. Following the pragmatic deployment concept of the Grid, they should in principle be able to offer the required services in environments which implement only a single prioritized per-hop behavior: the Expedited Forwarding per-hop behavior.

A consequence of this is the requirement to incorporate traffic engineering capabilities into the bandwidth broker design. Traffic engineering is a vehicle for the provision of per-domain behaviors, i.e. a basic building block for offering strict service parameters for aggregate scheduling. Additionally, it allows the controlled isolation of a particular set of service requests and can thus support the path-differentiation of services. Feedback mechanisms which ensure the consistency between the admission control and the actual path of a flow in the network are a requirement to pertain IP- and service-robustness.

From an application point of view, bandwidth brokers should support the effective use of the network resource. This leads to the demand to incorporate feedback mechanisms which support the development of adaptive applications. By either adapting the injected traffic profile or the service request, applications can iteratively optimize their use of the networking resource. For TCP-based applications using an assigned Guaranteed Rate tunnel this is, however, an unsatisfying approach. The protocol specific flow and congestion management mechanisms of TCP decouple the control of the application about the injected traffic profile. An embedded approach, in which the bandwidth broker is capable of pacing the injected traffic appropriately to the capacity of the tunnel is the more reasonable approach for this type of application.

Chapter 5

Design of a Grid Bandwidth Broker

This chapter evolves a flexible bandwidth broker architecture that addresses the particular design of emerging Grid applications.

5.1 Design Layout

This section structures the tasks of a bandwidth broker and describes the fundamental building blocks of the proposed framework.

5.1.1 Scope of Control

The task of a bandwidth broker is to facilitate and control the access to network services for applications. Here, the bandwidth broker performs admission control, resource provisioning and other policy decisions, and can thus be viewed as resource manager of the network. From the Differentiated Services (DS) architecture point of view, a bandwidth broker expands the Per-Hop Behavior (PHB) of aggregates to a Per-Domain Behavior (PDB) [95] (refer to Section 3.2.2 for a definition).

Each PDB has measurable, quantifiable, attributes that can be used to describe the service characteristics of the traffic. While the associated PHBs do have a great impact on the achievable PDB attributes, the topology and the injected traffic profile influence the service as well.

Because of the fact that end-to-end guarantees in Grid environments are likely to happen in complex network environments where multiple independent administrative organizations are responsible for the operation of subparts of the network, it is very unlikely that a single

bandwidth broker will control more than one administrative domain. Instead, each administrative domain wishes to have control over their resources and will thus operate its own policy decision point. A network reservation for traffic traversing multiple domains must therefore obtain multiple network reservations, as shown in Figure 5.1.

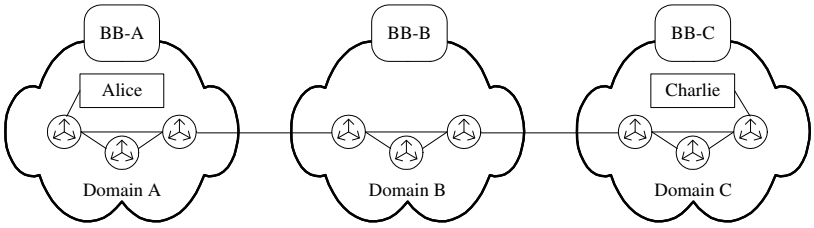

Figure 5.1: The multi-domain reservation problem. Alice needs to contact three bandwidth brokers (BB-A, BB-B, BB-C) to make a network reservation from her computer in domain A to Charlie's computer in domain C. From the perspective of the reservation of a Grid resource, the problem can either be handled by actively requesting a reservation in each domain, or, transparently for Alice, by contacting BB-A which is responsible for propagating the reservation requests to the bandwidth broker of the downstream domain.

Whenever service guarantees have to be obtained in multiple domains, a specific contract between peered domains comes into place. This contract is called a Service Level Agreement (SLA). It regulates details about available service levels by specifying the related parameters, access policies, and associated costs. Hence, the SLA regulates the acceptance and the constraints of a given traffic profile. Note that service associated costs and access rules are also listed in this contract. In contrast to the formal character of an SLA, Service Level Specifications (SLS) are used to technically describe the appropriate QoS parameters [128] that an SLA demands. According to Nichols and Carpenter [95] the definition of an SLS is as follows:

Definition 8 *A Service Level Specification (SLS) is a set of parameters and their values which together define the service offered to a traffic stream by a DS domain. It is expected to include specific values or bounds for Per-Domain Behavior parameters.*

SLSs are used by bandwidth brokers as input information for their admission control procedures. End-to-end guarantees can then be built by a chain of SLSs, i.e. by the transitivity of all SLSs.

An example SLS could list domain and service specific boundaries for delay and jitter. It could also specify a minimum bandwidth available for requesters, depending on either their source and/or destination address, and a threshold for the available bandwidth. In this context, the service requester would be able to claim the demand of bandwidth dynamically. Whenever SLSs allow a service provision independently from the actual egress point, the granularity of control is quite limited:

- The maximum available bandwidth for this type of service does only depend on the minimum link capacity and not on the actual path in the network
- Multiple ingress domains might interfere and thus might reduce the amount of available bandwidth again
- Delay and jitter boundaries used in the SLA have to be calculated based on worst-case assumptions for their use of network links

Because of these limitations, this dissertation proposes an architecture where an SLS lists the service parameters between each pair of ingress- and egress point. In this environment, a bandwidth broker dynamically maps service requests to their traversing path. A service request is only granted if the bandwidth broker is able to validate the service availability on all traversed links. Each service request must therefore contain a destination address which allows the derivation of the related destination domain.

The desired granularity of service control is for transient domains different than for end-domains. While end-domains typically perform their admission control on a per-request base, transient domains are focusing on their contractual conformance. The more dynamic contractual obligations are, the more state updates have to be processed. It is likely to happen that dynamic SLAs are also formulating rules which regulate the dynamic state changes. An example for this is to claim that reservation requests have to be made in chunks of a particular granularity. This is reasonable as transient domains might want to reduce their signaling overhead, and, in applying traffic engineering mechanisms, they might want to use these reasonable sized reservations to map them efficiently to the topology. This leads to the abstraction of a core tunnel:

Definition 9 *A core tunnel is an aggregated uni-directional reservation between the two end-domains. It connects egress router of the source-domain with the ingress router of the destination-domain by means of the service request parameters. Any subject authorized to use this tunnel may then request portions of this aggregate bandwidth by contacting just the bandwidth brokers of relevant end-domains. Intermediate domains do not need to be contacted as long as the total bandwidth remains less than the size of the tunnel.*

A core tunnel instantiation of a single transient domain nicely relates to the Per-Domain Behavior and fits well to the proposal to incorporate traffic engineering mechanisms during its allocation. In some sense, the PDB can be viewed as a core tunnel with particular service parameters. Core tunnels traversing multiple domains can then be viewed as the concatenation of the PDBs of all transient domains.

5.1.2 Basic Architectural Framework

A first functional decomposition of a bandwidth broker is described in [23]. This thesis proposes a different, more integrated decomposition of building blocks:

External Signaling Interface: This block is responsible for the receipt and the propagation of service request relevant messages. It interfaces with external entities including the Grid resource management system. Requests can either be originated directly by an application which uses intra-domain specific authentication and authorization mechanisms, by a peered-upstream bandwidth broker, or by an upstream bandwidth broker which is responsible for signaling service requests to the tail-end of a core tunnel.

State Repository: This block is responsible for the bookkeeping of reservations. It incorporates the knowledge about the link topology of the domain which includes any routing information. Reservations might be of different states and might consist of individual micro-flows or aggregates. The topology information is typically derived by a link state protocol and can change over the time. To preserve IP robustness, the state repository must follow a soft state model. Hence, integrating the state repository with the information gathered by network management tools, active monitoring techniques, and by routing protocols is a design goal. Whenever directory enabled networks are used, the state base should be incorporated with the directory.

Policy Repository: This block is the clearinghouse for admission control relevant rules. In addition to local intra-domain rules, it also denotes the service level agreements with peered domains.

Internal Signaling Interface: This block is responsible for interfacing with the routers. It propagates the current QoS policy information, monitors the edge-devices for reservation relevant events, and supervises the domain for events related to the state or policy repository.

Control Procedures: This set of procedures is driven by the signaling interfaces. Using the repositories, it processes reservation relevant events and initiates the appropriate reaction such as initiating a configuration update through the internal signaling interface or responding to the requester through the external signaling interface.

Figure 5.2 illustrates the proposed architecture.

5.1.3 External Signaling Interface

There are two different types of environments bandwidth brokers are used in:

1. end-domains
2. transient-domains

As explained above, end-domains typically want a fine granular control about the access to their network services, which often means the authentication and authorization of each

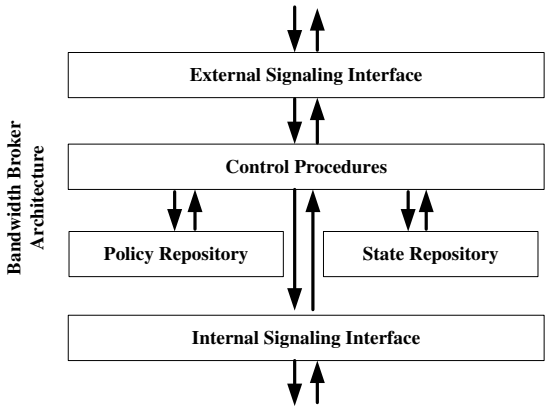

Figure 5.2: Functional decomposition of a bandwidth broker which consists of five basic building blocks.

single service request. In the simplest case, the source domain is also end-domain which means that the service request is domain internal and only handled by a single bandwidth broker. Here, the external signaling interface is responsible for the interaction between user and bandwidth broker. From the perspective of a Grid resource management framework, end-domain bandwidth brokers operate as local resource managers. Hence, they are accessed by standard resource acquisition methods, originated either by end-users or higher-level services. Whenever the underlying security infrastructure uses the principle of delegation, the higher-level service impersonates its requester which leaves the trust relation ship to that received in the case when the user was communicating directly with the bandwidth broker.

Transient-domains are focusing on their contractual obligations, i.e. on their ability to fulfill all SLAs in an economical manner. Policies and procedures are therefore likely to be different from that of end-domains. While end-domains typically grant access on a per-user and per-request base, transient-domains might want to regulate that access to a specific service is only granted in chunks of a particular size, i.e. bandwidth units and service time length, to avoid frequent updates of the edge devices. Static SLAs are the extension of these limitation, as they specify that the service is allocated at once, and is available for any time the SLA specifies. On the other hand, transient-domains are quite interested in service relevant events of their downstream domains as this might affect their own ability to serve their customers. The ability to perform asynchronous event notification is essential to address this demand. From the perspective of a Grid resource management system, these heterogeneous policy requirements can be addressed by two different concepts:

- The network is partitioned into domains. Hence, the networking resource is controlled by a set of bandwidth brokers which are the local resource manager of their particular domain. A network reservation is then comparable to a coordinated allocation of multiple resources. Any requester has to contact all relevant bandwidth

brokers explicitly and is thus responsible for delivering the related policy information. There is a direct trust relationship between requester and bandwidth broker.

- The network is viewed as a single resource managed by the bandwidth broker of the source domain. Downstream domains are hidden to the requester. This encapsulation is valid because end-to-end services always depend on the aggregation of the achievable services in each relevant domain. This concept, however, requires that service requests are transparently signaled between the relevant bandwidth brokers among the path.

Both design options are explicitly discussed in Section 5.4.2.

In addition to the design decision whether the network is handled as a single resource or not, the external signaling interface should also be viewed as a separate, not necessarily Grid related interface. By decoupling the interface from the Grid resource manager interface, the architecture gains the flexibility to evolve both parts more or less independently. This dissertation therefore proposes to embed an additional layer between the Grid resource management and the external signaling interface. This layer implements a separate process which is a wrapper between the Grid resource management semantics and the semantics interfaces the external signaling interface.

It is clear that the interface should implement the authentication method required in the related scenarios. Though it is decoupled from the Grid resource management, it still requires a flexible authorization framework. A generic authorization framework for the external signaling interface is evolved in Section 5.4.2.

5.1.4 State Repository

The State Repository is responsible for the bookkeeping of reservations. This dissertation proposes a simple slot-table [30, 72] model where all reservations are placed in a co-ordinate system based on the start- and end-time (x-axis) and their requested amount of the service class (y-axis). The latter will typically be bandwidth, either in a percentage form or in an absolute value. Reservations are thus stored in forms of rectangles in a co-ordinate system. Whenever a reservation arrives, it must be able to place the related rectangle in the co-ordinate system without any overlap, otherwise the reservation exceeds the dedicated service amount for at least a small period of time. Figure 5.3 illustrates the graphical representation of slot tables.

Though more efficient data structures such as segment trees have been proposed [116], their basic functionality is equivalent. The model of a slot table with its nice graphical representation is convenient to describe the required functionality of the state repository, regardless whether the implementation does actually rely on different data structure. Aggregation of reservations is the conceptual vehicle to address scalability problems of the

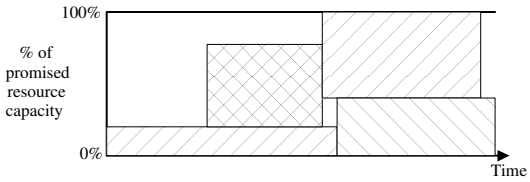

Figure 5.3: The State Repository uses slot tables to keep track about capacity promises. Here, the graphical representation of a slot table is illustrated in which four reservations were placed.

state repository. The State Repository extends the slot table model in such a way that a bandwidth broker maintains a slot table per link. Every reservation request must somehow be consistently mapped to the related path in the topology. It is important to note that IP-robustness requires that this mapping might vary over the time.

To support this dynamic mapping, an additional abstraction is introduced which we call "a traffic trunk". A traffic trunk connects ingress and egress router with a particular level of service. Hence, traffic trunks are the persistent representation of an accepted service request. In the simplest case, which is typically existent in end-domains, each single uni-directional reservation is related to a traffic trunk. The only difference between a traffic trunk and the reservation for the particular flow is that trunks may exist without having to denote the end-points, i.e. without listing both port numbers. Traffic trunks are a vehicle for the support of advance reservation where any missing reservation related information is completed when the service is claimed, i.e. when the port numbers are known. In the more complex case of a transient domain, a trunk is typically equivalent to a core tunnel reservation, i.e. an aggregated reservation which might be used by multiple flows. Note that a trunk is always uni-directional. It only provides guarantees for packets traversing from the source downstream to the destination. Any upstream guarantees have to be requested separately.

For the State Repository the introduction of a traffic trunk means that it has to persistently store their attributes. Trunks represent the unit of interest for bandwidth brokers, as they represent the granularity of interest in both environments: end- and transient-domains. Network reservations are always represented by traffic trunks. Whenever the reservation is instantiated or the State Repository is updated, traffic trunks are dynamically mapped to the topology using an admission control procedure with incorporated traffic engineering mechanisms.

While a traffic trunk itself is dynamically mapped to the topology, the proposed architecture assumes that the end-points of any traffic trunk are fix. These fix points are the only way to assure that multi-domain reservations are instantiated by the correct domains. Whenever a trunk would end at a different end-point, this would either be a different ingress or egress router. The problem here is that a traffic trunk could direct to a peered domain without any activated reservation. This, of course, does have an impact on the requirements for the State

Repository. The knowledge of routing updates is not just important for a consistent admission control model, it is also essential for the ability to setup guarantees between domains. For transient domains, traffic trunks can be viewed as a tunnel which connects the egress router of the peered upstream domain with the ingress router of the peered downstream domain.

5.1.5 Policy Repository

Whenever a service request arrives, a bandwidth broker uses a domain and requests specific set of rules to decide whether the request can be granted or not. This set of rules includes

- Requester relevant information (who is allowed to access a specific service)
- Request relevant information (when and what is a specific entity allowed to use)
- Domain wide policy information (are there specific domain wide rules such as service access is only granted in chunks of 5 Mbps)
- SLS relevant rules (what are the principal capabilities the bandwidth broker can claim at the downstream domains)
- Network management derived information (what are the specific capabilities of the network devices and of the management instantiation)

The above listed set of rules indicates that there is a fundamental difference between the policy repository of transient domains and that of end-domains. End-domains of network reservations will often want to grant access to their added-value network services based on attributes that precisely describe the requester and their authorization to actually produce or consume the traffic. For example, an end-domain might enforce authorization policies stating that network reservations are accepted only if the requester is capable of presenting authorization attributes that fulfill listed requirements associated with the resource, possibly including a track record of past use. Alternatively, it may be sufficient that that the requesting subject is a known subject in the end-domain, and access is granted without considering rules concerning the requested resource. Transient domains, however, are more concerned about fulfilling their contracts with their peered domain. Hence, they typically do not care about individual subjects requesting services, but on requests originating from a well known subject, i.e. a peered bandwidth broker. The consequence is that the basic architecture does not make assumptions about the form of authorization rules. Instead, the model proposes a module which simply validates the authorization on a per-request base. As a prerequisite for this, it requires that all requests are mutually authenticated, i.e. the bandwidth broker and the requester are capable of determining the corresponding entity.

Domain wide policy information and network management derived information describe the specific capabilities of the underlying topology. This includes the specification of link

capacities, traffic engineering rules, and service provision capabilities of the network devices.

SLS capabilities connect the agreements with peered upstream domains with that of peered downstream domains. Whenever requests arrive which cannot be met by a downstream SLS, it cannot be handled.

5.1.6 Control Procedures

The core of a bandwidth broker is a set of routines which process service requests with respect to the constraints given by the repositories. In addition to this task, adaptive techniques should also be supported. An example for this is the selective propagation of reservation relevant events.

To structure the set of control procedures, the following functional decomposition can be used:

- Request processing routines which process requests originated from end-users and upstream bandwidth brokers
- Response processing routines which process messages originated by peered downstream bandwidth brokers
- Event processing routines which process asynchronous messages received by one of the signaling interfaces. Note that the events can also be caused by a polling or timer strategy of a particular control procedure. An example for the latter is the set up of a timer which indicates when the starting time of a reservation has arrived.

5.1.7 Internal Signaling Interface

This building block of a bandwidth broker interfaces with networking devices. In a DS environment this mainly, but not exclusively, includes the edge routers. The internal signaling interface is designed to assure the consistency between the state of the network and the State Repository. Whenever a router state changes, either by some automatism such as an routing protocol update, or by some other event such as an administrative configuration change of a particular edge router, the internal signaling interface is responsible for the receipt of the event and its bandwidth broker internal propagation.

The proposed model does not rely on a particular mechanism of the internal signaling interface. Of course, the use of the outsourcing model of the Common Open Policy Service (COPS) protocol [34] appears to be a good mechanism for the asynchronous propagation of events. In this model, routers, also called Policy Enforcement Points (PEP), are able

to asynchronously inform a Policy Decision Point (PDP), in our context the bandwidth broker, about events. COPS defines a generic message for these events. The actual reasons for such an event depends on the underlying capabilities of the network devices. A typical example of what the PEP is capable of is that it automatically propagates its actual QoS configuration to the PDP when it is restarted. This information is quite important, as it ensures that the State Repository of the bandwidth broker can be kept consistent to the actual configuration of the routers.

However, the proposed architecture does not rely on COPS. The framework assumes the basic functionality of experiencing state updates without binding this to a particular mechanism. An alternative instantiation of the internal signaling interface could also follow a pull-model. In this scenario, the bandwidth broker periodically connects to the edge routers and extracts the related state information. In contrast to the asynchronous COPS outsourcing model, this implementation would not allow a fine granular consistency, but it could provide more flexibility with a reasonable delay for the propagation of state changes.

5.2 Support of a Coordinated Reservation and Allocation

The general problem of QoS implementation and management is receiving increased attention (see, e.g. [62]). Proposals for advance reservations typically employ cooperating servers that coordinate advance reservations along an end-to-end path [136, 42, 30, 64]. Techniques have been proposed for representing advance reservations, for balancing immediate and advance reservations [42], for advance reservation of predictive flows [30]. However, this work has not addressed the co-reservation of resources of different types, as it is required in Grid environments. This section refines the fundamental of the design listed in Figure 4.1 by embedding the bandwidth broker into the Grid advance reservation context as it is illustrated by Figure 5.4. The external signaling interface of the bandwidth broker is encapsulated from the Grid resource management system by a process, a wrapper, which intermediates between the two interfaces. Grid applications use a unique user interface, consisting of a set of well defined operations and hierarchically structured service request attributes to interface with the bandwidth broker.

The root-level contains the following generic reservation attributes:

Start Time: The earliest time that the reservation may begin. A reservation always has a start time, even if it is an immediate reservation, which begins at the time the requester submits the request.

Duration: The time a reservation lasts

Service-Specific Parameters: Parameters that are unique to a specific service, such as a guaranteed bandwidth or delay boundaries for network reservations.

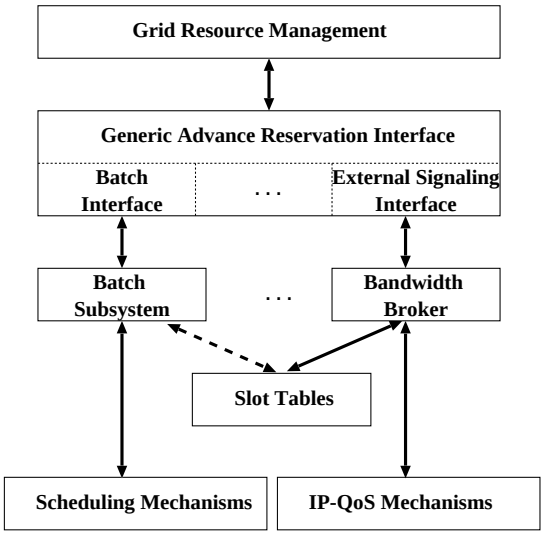

Figure 5.4: The integration of a bandwidth broker into the Grid resource management. The external signaling interface is encapsulated by a Generic Advance Reservation Interface which maps Grid requests to requests understood by a particular resource manager. Components of the bandwidth broker are useful for other resources as well. Slot tables, as proposed for an implementation of the State Repository, are a general abstraction which are also of interest for the development of other resource management systems.

Optionally, a reservation may specify the additional attribute “End Time”. The purpose of this optional parameter is to allow a more flexible placement of the actual reservation. If the difference between end and start time exceeds the value of duration, any given time subinterval of the correct duration within the start-end interval is accepted for the reservation.

The next level inherits the root-level parameters and extends them by request parameters for a single resource. Parameters here are unique for each type of resource, such as bandwidth for a network reservation and number of nodes for a computation reservation. For network reservations this includes the identification of the communication end-points, the long-term average transmission rate, the amount of bytes which can be sent in addition to the long-term transmission rate, and boundaries for delay and jitter.

In some environments it is necessary to differentiate further. An example is the provision of two services based on a single aggregate behavior. The proposed design therefore allows additional child-levels in which the service request is detailed further. For network services this includes the indication to perform deadline file staging, access to a Premium or a GR service, and the use of an existing aggregated reservation.

Reinhardt [107] discusses the impact of advance reservation on reservation protocols. He identifies three basic phases of an advance reservation system. During the negotiation phase users propagate their requirements to the advance reservation system. After interpreting the specification the system launches an admission control procedure which checks the availability of the requested capability for this particular user over the given time in-

terval. Once a reservation passed the admission control procedure, the intermediate phase begins. During this phase the user is able to renegotiate his request. This may include the cancellation of the reservation as well as an in- or decrease of the requested resource. Feedback mechanisms initiated by the advance reservation system are used during this phase to inform the user about events which could result into a failure to achieve the requested type of service. Finally, the channel usage phase begins, where the reservation is actually instantiated. It is left to the implementation whether the instantiation of a reservation is done automatically or whether an explicit notification message is required.

The proposed model reflects this phase model by means of its supported operations. First, it distinguishes between making a reservation and claiming it. When a reservation is requested, it requires the specification of the mandatory attributes listed above. Once the reservation request is accepted by the bandwidth broker, a so-called reservation handle is returned from the system. This is some abstract credential that uniquely identifies the reservation and can then be used to determine whether a subject is authorized to perform subsequent operations on the reservation, i.e. to renegotiate the request. Renegotiation is supported by event asynchronous notification. Once a reservation was accepted, the bandwidth broker informs interested and authorized entities about related events. To actually use the reservation, it has to be claimed actively.

Proposed Reservation Operations

Create Reservation: This operation requests a reservation with a particular start time and duration. It is the base for all further operations. Note that network reservations are uni-directional reservations.

Modify Reservation: This operation modifies an existing reservation. For instance, one can increase the bandwidth that has already been requested. A modification which reduces its requirements normally succeeds. There may be factors that cause reduction modifications to fail, such as local policy that does not allow small reservations on some resources. However, the underlying implementation should implement this operation following the ”make before break” principle, i.e. by not cancelling the existing reservation and then making a new reservation.

Cancel Reservation: This operation terminates an existing reservation.

Bind Reservation: When the application is ready to use a reservation, it may need to provide run-time information that was not available at the time the reservation was made. This is known as binding a reservation. For example, network reservations require port numbers to be specified, but they are not usually known at reservation time. Not all reservations require such run-time parameters.

Unbind Reservation: A reservation can be unbound. It will no longer be usable by the subject using the reservation. It can then be rebound with new run-time parameters.

Committing Reservation: When a reservation is created, it can be specified as a two-phase commit reservation. Such reservations time-out after a specified time period, unless the reservation is committed.

Query Reservation Status: This operation returns the current status of the reservation. The status includes whether the start time of the reservation has been passed and whether the reservation has been claimed.

Query Reservation Attribute: This operation returns attributes associated with an existing reservation. This includes begin and end time of the given reservation, and whether it is a two-phase commit reservation. It also includes specific information required to actually use a bound reservation. Example attributes are a directory name where data was staged on, or a queue name which has to be used for submitting a job.

Register Callback: This operation registers a function that will be called when the status of a reservation changes or when the reservation manager wishes to provide extra information to the application.

When a reservation is made, a Control Procedure is proving the authorization of the requesting subject. Once the basic authorization check was successful, an admission control procedure approves whether the bandwidth broker willing to delegate the request resource capability to the requester during the specified time constraint. To perform this check, it might use additionally given policy information, either passed within the request, or acquired at a trusted entity, i.e. a policy server. While this admission control step definitively involves the access to local data structures which are supposed to keep track of scheduled reservations, it could also be a more complex procedure, involving admission control checks in downstream domains. The latter is the case when an application is trying to reserve end-to-end network bandwidth in a multi-domain environment, and the network is handled as a single Grid resource. In this scenario, the admission control has to be done at each traversed domain and the answer of the check is the conjunction of all answers. Consequently, the admission control tests of a distributed bandwidth broker architecture typically involves complex signaling mechanisms.

5.3 Building Per-Domain Behaviors

The creation of a core tunnel in transient domains can be interpreted as an agreement to serve the related aggregate with a particular service level, i.e. a Per-Domain Behavior. So far, Section 3.4 presented the basic formalism for calculating a boundary for the delay: the network calculus. However, the discussion focused on the introduction of a formal model for modeling network capabilities for single flows. This section extends this model to aggregates. In contrast to [9], the proposed framework does not assume an arbitrary

topology. Instead it incorporates the ability to perform traffic engineering and deduces the formalism for determining an assured delay boundary for an LSP candidate.

A DS implementation of a Premium service often relies on the ability to serve the EF aggregate with non-preemptive Priority Queuing (PQ). Assuming a token bucket constrained arrival curve $\alpha(T) = rT + b$, we receive a rate-latency service curve β with rate C and latency $\frac{L}{C}$ for PQ, where L denotes the amount of bits the non-preemptive interface queue is capable of storing.

Given an arrival curve and an extended service curve, the horizontal deviation gives a bound for the virtual delay (refer to Definition 2). Note that this relation does not assume a single network node. Instead, the applied formalism relies on the assumption of a system which serves packets with an extended service curve. Section 3.4 already stated that the service curve of a system consisting of multiple nodes $1..n$, where each of it provides a rate-latency service curve β_i, is given by $\beta_1 \otimes ... \otimes \beta_n$. By computing the end-to-end service curve as the min-plus convolution of the service curves of all nodes, the "pay bursts only once" principle establishes a better delay boundary than we would receive by sequentially modeling the domain node by node.

We now use this relation to formulate the service curve for a flow (or aggregate) in a domain. At the moment, we ignore any changes of the aggregate composition. For $1 \leq i \leq n$ let C_i denote the output link rate of node i which is offering a rate-latency service curve $\beta_{C_i,\frac{L}{C_i}}$. We are interested in the service curve of a Premium flow which is traversing the nodes $f(i) : 1 \leq i \leq I$, where $f : 1..I+1 \mapsto 0..n$ is a map which orders the traversed nodes. Note that we introduce an imaginary node $I+1$ which is the ingress router of the peered upstream domain. Hence, we receive $f(i) \neq f(j)$ for $i \neq j$ and $f(i) = 0 \Rightarrow i = I+1$. Following the above formula for the service curve of a concatenation of nodes, we receive:

$$\beta_{1..I}(t) = min_{1 \leq i \leq I}(C_{f(i)})[t - \sum_{i=0}^{I} \frac{L}{C_{f(i)}}]^+ \tag{5.1}$$

We now have a formula for expressing the service curve of a system consisting of multiple nodes each of it offering a rate-latency service curve. What is missing, however, is the transition to an aggregate service curve in which we consider multiplexing and de-multiplexing points.

What we additionally have to take into account is the fact that the traffic profile of the aggregate changes with every flow joining or leaving the aggregate. Let F_k be the flow of interest. Let us further assume that F_k is entering the aggregate at node $f(1)$ and is leaving it at $f(I)$. Note that we again assume an imaginary node $I+1$ which is the ingress router of the peered upstream domain. Hence, $f(I+1)$ is defined. We are interested in modeling the aggregate behavior in a way which enables us to derive boundaries for the delay and

the backlog at a particular time interval $[t_0, t_1]$ in which the amount of relevant micro-flows in the aggregate is constant. Let N be this number and let name all flows by $i \leq N : F_i$. Furthermore, we are interested in formulating rules for controlling the boundaries when the aggregate changes.

We start in formulating the problem for node $f(1)$ where we do have fresh arrival curves and assume that our domain is configured to use non-preemptive priority queuing between aggregates. Within a single aggregate packets are served in a FIFO manner.

Let $r_{f(1)}^{f(2)} \leq N$ be the amount of aggregate micro-flows which are traversing from node $f(1)$ to node $f(2)$, i.e. over the same link as F_k, and $m_{f(1)}^{f(2)} : 1..r_{f(1)}^{f(2)} \mapsto 0..N$ be a map which maps these micro-flows to their notation $F_{m_{f(1)}^{f(2)}(i)}$. If all micro-flows of the aggregate are constrained by some token bucket $(\rho_{m_{f(1)}^{f(2)}(i)}, \sigma_{m_{f(1)}^{f(2)}(i)})$, we receive an affine arrival curve for the whole aggregate which is constrained by a token bucket $(\sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)}, \sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)})$.

Because of the fact that all nodes are serving the aggregate with a rate-latency service curve β_i, i.e. $\beta_i(t) = C_{f(i)}[t - \frac{L}{C_{f(i)}}]^+$, we can derive an aggregate bound for the virtual delay:

$$d_{f(1)}(t) \leq \frac{L + \sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)}}{C_{f(1)}}$$

To derive a result for node $f(2)$, we have to construct an arrival curve for the exit traffic of node $f(1)$. Fortunately, Theorem 3 gives us a constraint for the output flow of node $f(1)$. As we are interested in formulating an arrival curve, we can assume $t \geq 0$ and $\sum_{i \in F_m} \rho_{m_{f(1)}^{f(2)}(i)} < C_{f(1)}$. Using the notation $\vee$ for $\sup$ or, if it exists, for max: $a \vee b = \max\{a, b\}$, the constraint is given by [14]:

$$\alpha^*(t) = (\sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + t \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)}) \oslash (\beta_{f(1)}(t))\ (t) \tag{5.2}$$

$$= \sup_{u \geq 0} \{\sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + (t+u) \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)} - C_{f(1)}[u - [\frac{L}{C_{f(1)}}]^+\} \tag{5.3}$$

$$= \sup_{0 \leq u \leq \frac{L}{C_{f(1)}}} \{\sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + (t+u) \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)} - C_{f(1)}[u - \frac{L}{C_{f(1)}}]^+\} \vee \tag{5.4}$$

$$\sup_{u > \frac{L}{C_{f(1)}}} \{\sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + (t+u) \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)} - C_{f(1)}[u - \frac{L}{C_{f(1)}}]^+\} \tag{5.5}$$

$$= \sup_{0 \leq u \leq \frac{L}{C_{f(1)}}} \{\sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + (t+u) \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)}\} \vee \tag{5.6}$$

$$\sup_{u > \frac{L}{C_{f(1)}}} \{\sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + (t+u) \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)} - C_{f(1)}[u - \frac{L}{C_{f(1)}}]^+\} \tag{5.7}$$

$$= \sup_{0 \leq u \leq \frac{L}{C_{f(1)}}} \{\sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + (t+u) \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)}\} \vee \tag{5.8}$$

$$\sup_{u > \frac{L}{C_{f(1)}}} \{\sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + (t+u) \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)} - C_{f(1)}u + L\} \tag{5.9}$$

$$= \{\sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + (t + \frac{L}{C_{f(1)}}) \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)}\} \vee \tag{5.10}$$

$$\sup_{u > \frac{L}{C_{f(1)}}} \{\sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + (t+u) \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)} - C_{f(1)}u + L\} \tag{5.11}$$

By using the information about u we can now calculate the remaining $\sup$-value:

$$\alpha^*(t) = \{\sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + (t + \frac{L}{C_{f(1)}}) \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)}\} \vee \tag{5.12}$$

$$\sup_{u > \frac{L}{C_{f(1)}}} \{\sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + t \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)} + u(\sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)} - C_{f(1)}) + L\} \tag{5.13}$$

$$= \{\sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + (t + \frac{L}{C_{f(1)}}) \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)}\} \vee \tag{5.14}$$

$$\{\sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + (t + \frac{L}{C_{f(1)}}) \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)}\} \tag{5.15}$$

$$= \sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + (t + \frac{L}{C_{f(1)}}) \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)} \tag{5.16}$$

$$= \sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + L\frac{\sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)}}{C_{f(1)}} + (\sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)})t \tag{5.17}$$

We thus receive a token bucket constraint for the exit traffic of node $f(1)$. The average rate of the exit aggregate is the summation of all aggregate micro-flows, but we experience an increase in burstiness of

$$L\frac{\sum_{i=1}^{r_{f(1)}^{f(2)}} (\rho_{m_{f(1)}^{f(2)}(i)})}{C_{f(1)}}$$

To incorporate the advantage of the "pay bursts only once principle", we concatenate the rate-latency service curve of all nodes. So for $v > 1$ let node $f(v)$ be the node at which the composition of the aggregate is changing the first time. We receive:

$$\alpha^*(t) = \sum_{i=1}^{r_{f(1)}^{f(2)}} \sigma_{m_{f(1)}^{f(2)}(i)} + \sum_{j=1}^{v-1} \frac{L}{C_{f(v)}} \sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)} + (\sum_{i=1}^{r_{f(1)}^{f(2)}} \rho_{m_{f(1)}^{f(2)}(i)})t \tag{5.18}$$

By reapplying this procedure at downstream nodes we receive a consistent end-to-end model for aggregates. What is missing, however, is a model for de-multiplexing the aggregate. Here, the formalism must be able to split the arrival curve into two parts: the relevant part and the part which is taking a different path in the next node. The basic building blocks

here are the associativity and the commutativity of the Min-Plus convolution which allow to decompose the arrival curve of each input interface to a set of flows—which is called sub-aggregate—which will exit the system on a common output link. Figure 5.5 illustrates how this decomposition can be performed, but it also demonstrates that aggregates can also be de-multiplexed at every node. Whenever there are micro-flows which exit on a different output interface than the flow of interest, they are not part of the relevant input constraint and thus not part of the sub-aggregate we will consider. Assume that we are interested in providing a delay boundary for flow F_2 in Figure 5.5. For node 1, we have to use the arrival curve constrained by the arrival curves of F_2, F_5, F_6, F_7. For node 2, however, we are focusing on the arrival curves of flows F_2, F_5, F_8, F_{11}. To calculate the arrival curve in node 2, we thus have to determine the service curve for the sub-aggregate of F_2, F_5 of node 1, i.e. we have to separate our original aggregate in node 1 to the set of flows which will take the same path than our reference flow in node 2 and those taking a different path. Fortunately, we can assume that the service curve of a non-preemptive priority queuing node is serving the packets in FIFO order. The following two theorems [14] give us the mechanism to model the fragmentation of aggregates:

Theorem 4 Service curve of sub-aggregates for FIFO schedulers *Consider a node service two flows or sub-aggregates in FIFO order. Assume that flow f_i is constrained by one some token bucket (ρ_i, σ_i). Assume the node guarantees to the aggregate of the two flows a rate-latency service curve $\beta_{R,T}$. If $\rho_1 + \rho_2 < R$, then the flow 1 has a service curve equal to the rate-latency function with rate $R - \rho_2$ and latency $T + \frac{\sigma_2}{R}$.*

Theorem 5 Output constraint of sub-aggregates for FIFO schedulers *Consider a node service two flows or sub-aggregates in FIFO order. Assume that flow f_i is constrained by one some token bucket (ρ_i, σ_i). Assume the node guarantees to the aggregate of the two flows a rate-latency service curve $\beta_{R,T}$. If $\rho_1 + \rho_2 < R$, then the output of the first flow is constrained by a token bucket with parameters (ρ_1, σ_1^*) with*

$$\sigma_1^* = \sigma_1 + \rho_1(T + \frac{\sigma_2}{R})$$

To apply Theorem 5 at core routers we have to identify the arrival constraints for the relevant and the not relevant flows. While Formula 5.18 gives us a constraint for the whole aggregate traversing a set of nodes, we now have to distinguish between two sets of flows:

1. those flows which stay in the aggregate at the peered downstream node
2. and those which are de-multiplexed at the peered downstream node.

Again we start with a simple model and assume that the aggregate is fragmented at node $f(2)$. Let $r_{f(1)}^{(f(2),f(3))} \geq 1$ denote the amount of flows in node $f(1)$ which are sharing the output link in node $f(2)$, i.e. the amount of flows passing $(f(1), f(2), f(3))$. Let

us also denote the amount of flows which are de-multiplexed at node $f(2)$, i.e. flows which are passing $(f(1), f(2))$, but not $f(3)$ as $R_{f(1)}^{(f(2),f(3))}$. We now define two maps: $m_{f(1)}^{(f(2),f(3))} : 1..r_{f(1)}^{(f(2),f(3))} \mapsto 0..N$ and $M_{f(1)}^{(f(2),f(3))} : 1..R_{f(1)}^{(f(2),f(3))} \mapsto 0..N$ which maps these micro-flows to their notation $F_{m_{f(1)}^{(f(2),f(3))}(i)}$ resp. $F_{M_{f(1)}^{(f(2),f(3))}(i)}$. Using Theorem 5 we obtain a new arrival constraint which can be used in the downstream node by

$$\alpha_{f(1)}^{(f(2),f(3))}(t) = \sum_{i=1}^{r_{f(1)}^{(f(2),f(3))}} \rho_{m_{f(1)}^{(f(2),f(3))}(i)} t + \sum_{i=1}^{r_{f(1)}^{(f(2),f(3))}} \sigma_{m_{f(1)}^{(f(2),f(3))}(i)} + \tag{5.19}$$

$$L \frac{\sum_{i=1}^{r_{f(1)}^{(f(2),f(3))}} \rho_{m_{f(1)}^{(f(2),f(3))}(i)}}{C_{f(1)}} + \tag{5.20}$$

$$\sum_{i=1}^{r_{f(1)}^{(f(2),f(3))}} \rho_{m_{f(1)}^{(f(2),f(3))}(i)} \frac{\sum_{i=1}^{R_{f(1)}^{(f(2),f(3))}} \sigma_{M_{f(1)}^{(f(2),f(3))}(i)}}{C_{f(1)}} \tag{5.21}$$

By de-multiplexing the arrival constraint for node $f(2)$, we effectively increased our burstiness by two addend:

1. The burst caused by the delay in node $f(1)$. Within this time scale packets of the relevant flows might arrive at rate $\sum_{i=1}^{r_{f(1)}^{(f(2),f(3))}} \rho_{m_{f(1)}^{(f(2),f(3))}(i)}$. A non-preemptive priority node will serve all of them before any other packet is served. We thus might receive an additional burst of relevant packets on the output link.

2. The possible burst caused by additional delay introduced by the flows which are leaving the relevant path in the next node. It might occur that the whole burst of packets of these flows arrive earlier than the relevant packets. Because of the FIFO property of our scheduler, these are served prior any packet of the relevant sub-aggregate is served. We thus have to add this additional delay to the time in which packets of the relevant flows can be queued in a row.

Restructuring the result we receive:

$$\alpha_{f(1)}^{(f(2),f(3))}(t) = \sum_{i=1}^{r_{f(1)}^{(f(2),f(3))}} \rho_{m_{f(1)}^{(f(2),f(3))}(i)} t + \tag{5.22}$$

$$\sum_{i=1}^{r_{f(1)}^{(f(2),f(3))}} (\sigma_{m_{f(1)}^{(f(2),f(3))}(i)} + \rho_{m_{f(1)}^{(f(2),f(3))}(i)} \frac{L + \sum_{j=1}^{R_{f(1)}^{(f(2),f(3))}} \sigma_{M_{f(1)}^{(f(2),f(3))}(j)}}{C_{f(1)}}) \tag{5.23}$$

The calculation of a constraint for any set of flows which is de-multiplexed is analogous $(j \neq f(3))$:

$$\alpha_{f(1)}^{(f(2),j)}(t) = \sum_{i=1}^{r_{f(1)}^{(f(2),j)}} \rho_{m_{f(1)}^{(f(2),j)}(i)} t + \sum_{i=1}^{r_{f(1)}^{(f(2),j)}} \sigma_{m_{f(1)}^{(f(2),j)}(i)} + L\frac{\sum_{i=1}^{r_{f(1)}^{(f(2),j)}} \rho_{m_{f(1)}^{(f(2),j)}(i)}}{C_{f(1)}} + \tag{5.24}$$

$$\sum_{i=1}^{r_{f(1)}^{(f(2),j)}} \rho_{m_{f(1)}^{(f(2),j)}(i)} \frac{\sum_{i=1}^{R_{f(1)}^{(f(2),j)}} \sigma_{M_{f(1)}^{(f(2),j)}(i)}}{C_{f(1)}} \tag{5.25}$$

Note that for $r_{f(1)}^{(f(2),f(3))} = r_{f(1)}^{f(2)}$ and thus $R_{f(1)}^{(f(2),f(3))} = 0$. Formula 5.21 provides the same result than before 5.17.

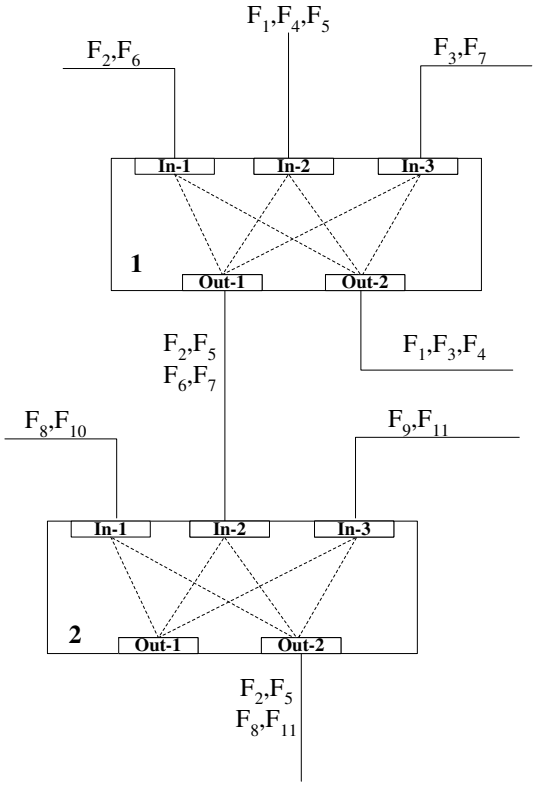

Figure 5.5: The constitution of an aggregate which is traversing through two nodes. Members of the aggregate arrive on three different input interfaces and exit on two different output interfaces. On each node, we will split the aggregate in two parts: the sub-aggregate which contains all flows using the same output link in the next node and the remainder.

Let us now illustrate Formula 5.21 for the scenario listed in Figure 5.5. Here, we are interested in the flows F_2, F_5. We thus have $r_1^{(2,3)} = 2$ and $R_1^{(2,3)} = 2$. Denoting the token

bucket constraint of flow F_i by (ρ_i, σ_i), $\alpha_1^2(t)$ is given by:

$$\alpha_1^{2,3}(t) = \sum_{i=1}^{2} \rho_{m_{f(1)}^{(2,3)}(i)} t + \sum_{i=1}^{2} \sigma_{m_1^{(2,3)}(i)} + L\frac{\sum_{i=1}^{2} \rho_{m_1^{(2,3)}(i)}}{C_1} + \sum_{i=1}^{2} \rho_{m_1^{(2,3)}(i)} \frac{\sum_{i=1}^{2} \sigma_{M_1^{(2,3)}(i)}}{C_1} \tag{5.26}$$

$$= (\rho_2 + \rho_5)t + \sigma_2 + \sigma_5 + L\frac{\rho_2 + \rho_5}{C_1} + (\rho_2 + \rho_5)\frac{\sigma_6 + \sigma_7}{C_1} \tag{5.27}$$

Note that we use Formula 5.21 also to formulate an arrival constraint for a single flow of the sub-aggregate. Suppose we want to formulate the constraint of flow F_2 between node 1 and 2 in the scenario of Figure 5.5. By applying the same formalism, but with a slightly modification of the related maps, we receive a token bucket constrained arrival curve of:

$$(\alpha_1^{2,3}(F_2))(t) = \rho_2 t + \sigma_2 + L\frac{\rho_2}{C_1} + \rho_2\frac{\sigma_5 + \sigma_6 + \sigma_7}{C_1} \tag{5.28}$$

We can use this information to calculate the delay boundary for flow F_2. The illustrated scenario offers two different ways to calculate a boundary:

- By the summation of the delay in each node. Note that due to the updated decomposition of the aggregates this model does not allow the use of the "pay bursts only once principle".
- By first deriving a domain-wide service policy for the end-to-end aggregate F_2, F_5 which is then de-multiplexed.

Calculation of the service curve for F_2 in each relevant node by applying Theorem 4:

$$\beta_{F_2}^1 = (C_1 - \rho_5 - \rho_6 - \rho_7)[t - \frac{L + \sigma_5 + \sigma_6 + \sigma_7}{C_1}]^+ \tag{5.29}$$

To formulate the service curve of node 2, we have to determine the arrival constraint for the flow F_5:

$$(\alpha_1^{2,3}(F_5))(t) = \rho_5 t + \sigma_5 + \rho_5\frac{L + \sigma_2 + \sigma_6 + \sigma_7}{C_1} \tag{5.30}$$

We thus have:

$$\beta_{F_2}^2 = (C_2 - \rho_5 - \rho_8 - \rho_{11})[t - \frac{L + \sigma_5 + \rho_5\frac{L+\sigma_2+\sigma_6+\sigma_7}{C_1} + \sigma_8 + \sigma_{11}}{C_2}]^+ \tag{5.31}$$

$$d_{F_2}(1,2) \leq d_{F_2}(1) + d_{F_2}(2) = \frac{L+\sigma_5+\sigma_6+\sigma_7}{C_1} + \frac{\sigma_2}{C_1-\rho_5-\rho_6-\rho_7} + \tag{5.32}$$

$$\frac{L+\sigma_5+\rho_5\frac{L+\sigma_2+\sigma_6+\sigma_7}{C_1}+\sigma_8+\sigma_{11}}{C_2} + \frac{\sigma_2}{C_2-\rho_5-\rho_8-\rho_{11}} \tag{5.33}$$

The alternative solution is to first determine the service curve for the end-to-end aggregate:

$$\beta^{(1,2)}_{F_2,F_5} = \beta^1_{F_2,F_5} \otimes \beta^2_{F_2,F_5} \tag{5.34}$$

$$= (C_1-\rho_6-\rho_7)[t - \frac{L+\sigma_6+\sigma_7}{C_1}]^+ \otimes \tag{5.35}$$

$$(C_2-\rho_8-\rho_{11})[t - \frac{L+\sigma_8+\sigma_{11}}{C_2}]^+ \tag{5.36}$$

$$= \min(C_1-\rho_6-\rho_7, C_2-\rho_8-\rho_{11})[t - \frac{L+\sigma_6+\sigma_7}{C_1} - \frac{L+\sigma_8+\sigma_{11}}{C_2}]^+ \tag{5.37}$$

We now use this aggregate service curve to derive a service curve for flow F_2:

$$\beta^{(1,2)}_{F_2} = (\min(C_1-\rho_6-\rho_7, C_2-\rho_8-\rho_{11}) - \rho_5) \tag{5.38}$$

$$[t - \frac{L+\sigma_6+\sigma_7}{C_1} - \frac{L+\sigma_8+\sigma_{11}}{C_2} - \frac{\sigma_5}{\min(C_1-\rho_6-\rho_7, C_2-\rho_8-\rho_{11})}]^+ \tag{5.39}$$

We can thus express a delay boundary by:

$$d_{F_2}(1,2) \leq \frac{L+\sigma_6+\sigma_7}{C_1} + \frac{L+\sigma_8+\sigma_{11}}{C_2} + \frac{\sigma_5}{\min(C_1-\rho_6-\rho_7, C_2-\rho_8-\rho_{11})} + \tag{5.40}$$

$$\frac{\sigma_2}{(\min(C_1-\rho_6-\rho_7, C_2-\rho_8-\rho_{11})-\rho_5)} \tag{5.41}$$

Compared to the result listed in 5.33 we avoided to pay for the bursts of F_5 multiple times.

To formalize the cascadation for a whole domain, we now assume a cycle-free feed-forward network. This assumption is applicable, as we are concerned about placing the aggregate into a set of uni-directional tunnels, each of it explicitly mapped to the underlying topology. By requiring the cycle-free property of tunnels, we receive a cycle free network. In doing so, we are able to iteratively use Formula 5.21. Note that we also showed slightly modified use of Formula 5.21 at Formula 5.28 which enabled us to formulate the token

bucket constraint for each individual flow. Hence, we can still assume that the cascadation will still result in token bucket constraints for individual flows.

Again we are concerned about some time interval $[t_0, t_1]$ where we can assume that the set of tunnels in the domain is constant. We model our domain by enumerating the networking nodes by $1 \leq d \leq n$. If we model the set of tunnels by $F_D = \{F_i : 0 \leq i \leq N\}$, we can formulate the amount of nodes traversed by tunnel F_i by n_i. We now define a family of functions $f_i : 0..n_i + 1 \mapsto 0..n_i$ which gives an ordered list of the traversed nodes for each tunnel F_i. Note that we extend each tunnel by two imaginary nodes 0 and $n_i + 1$ which are interpreted as the peered router in the peered domain. Hence, $f_i(j) = 0 \Rightarrow j = n_i + 1 \vee j = 0$.

Let F_a be the LSP of interest. Let us further assume that F_a is entering the domain at node $f_a(1)$ and is leaving it at $f_a(I)$. We now introduce the route interference number of the LSP, i.e. the number of multiplexing and de-multiplexing points, and denote it by RIN_a. Based on these relevant points, we define a map $\delta_a : 0..RIN_a + 1 \mapsto 0..I + 1$ which associates each point of interest to the LSP relative node, i.e. in each $f(\delta_a(j)$ is either another LSP joining or leaving the path of F_a. Note that we will use $\delta_a(0) = 0$ to indicate that the first point of interest is at the edge. We thus define $\delta_a(0) := \delta_a(1) - 1$, for $\delta_a(1) > 1$, and $\delta_a(0) = 0$ otherwise. Furthermore, we define $\delta_a(RIN_a + 1) = I$ for $\delta_a(RIN_a) < I$, and $\delta_a(RIN_a + 1) = I + 1$ otherwise.

To generalize the notation of $r_{f(1)}^{(f(2),f(3))}$, we assume to have the knowledge about the mapping of all LSPs in the domain. In doing this we also state that we are able to identify all points of interest δ_i. To improve readability, we abbreviate $a(i) = f_a(\delta_a(i))$. Based on this abbreviation we define $r_{a(1)}^{a(RIN_a)}$ as the amount of LSPs passing:

$$(f_a(\delta_a(1)), ..., f_a(\delta_a(RIN_a)))$$

For each $0 < i < RIN_a$, $R_{a(i)}^{a(i+1)}$ specifies the amount of LSPs which are traversing from node $a(i)$ to $a(i + 1)$, but are not passing node $a(RIN_a)$. In contrast to our procedure at the edge of the network we have to consider that the LSPs are not fresh anymore, i.e. they are not anymore constrained by the original token bucket. However, Theorem 5 and Formula 5.17 gives us the property that all arriving traffic is still constrained by some token bucket. For $1 \leq k \leq R_{a(i)}^{a(i+1)}$ let $(\overline{\sigma}_{a(i)}^{a(i+1)}(k), \overline{\rho}_{a(i)}^{a(i+1)}(k))$ be the related token bucket parameter. Note that these parameters have to be calculated for each individual node by deriving an arrival constraint for all related LSPs which all have a point of interest in the related node.

Now fix some node $f_a(\delta_a(d))$ with $0 \leq d \leq RIN_a + 1$.

In applying our formalism to the LSP using cascaded service curves for the nodes in between two points of interests, we receive an LSP service curve by the following recursive formulas:

For the special cases that either $\delta_a(0) = 0$, or $\delta_a(RIN_a + 1) = I + 1$, we define $\beta_a^0(t) = \beta_a^{I+1}(t) = \nu(t)$. Here, $\nu(t)$ denotes the neutral element of the min-plus convolution which is defined as $\nu(t) = 0$ for $t = 0$, and $\nu(t) = \infty$ for $t > 0$.

Whenever $\delta_a(0) > 0$, we receive the following service curves:

$$\beta_a^{\delta_a(0)}(t) = (\min_{1 \le i \le \delta_a(0)} C_{f_a(i)})[t - \sum_{i=1}^{\delta_a(0)} \frac{L}{C_{f_a(i)}}]^+ \tag{5.42}$$

For all $0 < s \le RIN_a$ we apply Theorem 4 and receive the following service curve:

$$\beta_a^{\delta_a(s)}(t) = (\min_{\delta_a(s-1)+1 \le i \le \delta_a(s)} (C_{f_a(i)} - \sum_{j=1}^{R_{a(s)}^{a(s+1)}} \overline{\rho}_{a(s)}^{a(s+1)}(j))) \tag{5.43}$$

$$[t - \sum_{i=\delta_a(s-1)+1}^{\delta_a(s)} \frac{L}{C_{f_a(i)}} - \frac{\sum_{j=1}^{R_{a(s)}^{a(s+1)}} \overline{\sigma}_{a(s)}^{a(s+1)}(j)}{\min_{\delta_a(s-1)+1 \le i \le \delta_a(s)}(C_{f_a(i)})}]^+ \tag{5.44}$$

Finally, the service curve for the remaining nodes $f_a(i)$ with $\delta_a(RIN_a + 1) \ge i > \delta_a(RIN_a)$ is given by:

$$\beta_a^{\delta_a(RIN_a+1)}(t) = \min_{\delta_a(RIN_a)+1 \le i \le \delta_a(RIN_a+1)} (C_{f_a(i)})[t - \sum_{i=\delta_a(RIN_a)+1}^{\delta_a(RIN_a+1)} \frac{L}{C_{f_a(i)}}]^+ \tag{5.45}$$

We now have the basic formalism for calculating the delay boundaries for a single LSP a, passing nodes $(f_a(1), ..., f_a(I))$. Note that it is necessary to list the LSP specific service curves of all nodes:

$$d_{(f_a(1),..,f_a(I))} \le h(\alpha_{f_a(0)}, \beta_a^{f_a(\delta_a(0))} \otimes ... \otimes \beta_a^{f_a(\delta_a(RIN_a+1))}) \tag{5.46}$$

Fortunately Formula 5.1 simplifies the calculation of the overall service curve: it is a rate-latency service curve.

$$\beta^*_{R',T'} = \beta_a^{f_a(\delta_a(0))} \otimes ... \otimes \beta_a^{f_a(\delta_a(RIN_a+1))} \tag{5.47}$$

$$R' = \min\{ \min_{1\leq i\leq \delta_a(0), \delta_a(RIN_a)+1\leq i\leq \delta_a(RIN_a+1)} (C_{f_a(i)}), \tag{5.48}$$

$$\min_{\delta_a(s-1)+1\leq i\leq \delta_a(s)} (C_{f_a(i)} - \sum_{j=1}^{R^{a(s+1)}_{a(s)}} \overline{\rho}^{a(s+1)}_{a(s)}(j))\} \tag{5.49}$$

$$T' = \sum_{s=1}^{RIN_a} \frac{\sum_{j=1}^{R^{a(s+1)}_{a(s)}} \overline{\sigma}^{a(s+1)}_{a(s)}(j)}{\min_{\delta_a(s-1)+1\leq i\leq \delta_a(s)}(C_{f_a(i)})} + \sum_{i=1}^{I} \frac{L}{C_{f_a(i)}} \tag{5.50}$$

And thus:

$$d_{(f_a(1),..,f_a(I))} \leq T' + \frac{\sigma_a}{R'} \tag{5.51}$$

When to merge LSPs?

Whenever a management system for LSPs uses explicit routing to map LSPs to a topology, the question about optimizing the placement of LSPs arises. Intuitively, multiplexing should be avoided in general. This is, of course, a very strong assumption. A more applicable scenario, however, is the question which of two possible (sub-)paths of the same length is better. Specifically, the question arises whether a necessary LSP multiplexing point should be placed closer to the ingress or to the egress node.

Whenever a bandwidth broker is requested to add an additional LSP b to a domain in which the path of an existing LSP a traversing the nodes $(f_a(1), ..., f_a(I))$ has to be merged at either node $f_a(1)$, or $f_a(I)$, the question arises about the delay boundaries for both LSPs, the one already existing and the new one. Note the assumption of merging either at node $f_a(1)$ or $f_a(I)$ can be made without loose of generality as we can easily construct a longer path for LSP a by

$$(f^*_a(1), ..., f^*_a(j-1), f_a(1), ..., f_a(I), f^*_a(j+1), ..., f^*_a(J))$$

Each node is offering a rate-latency service curve $\beta_{f_a(i)}(t) = C_{f_a(i)}[t - \frac{L}{C_{f_a(i)}}]^+$ and is serving packets in FIFO-order. Let the incoming traffic of LSP a be constrained by $\sigma_a + t\rho_a$ and the arriving packets of LSP b by $\sigma_b + t\rho_b$. Merging the path of both LSPs at node $f_a(1)$, i.e. b would pass the nodes $(f_a(1), ..., f_a(J))$ with $J > I$, we can model our system by a single system consisting of the nodes $(f_a(1), ..., f_a(I))$ which is offering a

service curve of:

$$\beta^*(t) = \min_{\{1 \le i \le I\}} C_{f_a(i)}[t - \sum_{i=1}^{I} \frac{L}{C_{f_a(i)}}]^+ \tag{5.52}$$

Applying Theorem 4 we receive a service curve for LSP a of:

$$\beta_a^*(t) = (\min_{\{1 \le i \le I\}} C_{f_a(i)} - \rho_b)[t - \sum_{i=1}^{I} \frac{L}{C_{f_a(i)}} - \frac{\sigma_b}{\min_{\{1 \le i \le I\}} C_{f_a(i)}}]^+ \tag{5.53}$$

Similarly, the service curve for LSP b is:

$$\beta_b^*(t) = (\min_{\{1 \le i \le I\}} C_{f_a(i)} - \rho_a)[t - \sum_{i=1}^{I} \frac{L}{C_{f_a(i)}} - \frac{\sigma_a}{\min_{\{1 \le i \le I\}} C_{f_a(i)}}]^+ \tag{5.54}$$

We now validate that we receive the same result we would receive by applying Formula 5.50. To apply the described scenario, we set $RIN_a = 1$, $\delta_a(0) = 0$, $\delta_a(1) = I$, and $\delta_a(2) = 0$. Hence we receive:

$$\beta^*_{R',T'} = \beta_a^{f_a(\delta_a(0))} \otimes ... \otimes \beta_a^{f_a(\delta_a(RIN_a+1))} \tag{5.55}$$

$$= \nu \otimes \beta_a^{f_a(I)} \otimes \nu \tag{5.56}$$

$$= \beta_a^{f_a(I)} \tag{5.57}$$

$$= (\min_{1 \le i \le I}(C_{f_a(i)} - \rho_b))[t - \sum_{i=1}^{I} \frac{L}{C_{f_a(i)}} - \frac{\sigma_b}{\min_{\{1 \le i \le I\}} C_{f_a(i)}}]^+ \tag{5.58}$$

LSP a is served by a rate-latency service curve again. Therefore the virtual delay of any packet in LSP a can be calculated by using Formula 5.51:

$$d_a^*(t) = \sum_{i=1}^{I} \frac{L}{C_{f_a(i)}} + \frac{\sigma_b}{\min_{\{1 \le i \le I\}} C_{f_a(i)}} + \frac{\sigma_a}{\min_{\{1 \le i \le I\}} C_{f_a(i)} - \rho_b} \tag{5.59}$$

The alternative solution is to merge LSP a at node $f_a(I)$. By assumption, LSP b is placed on a different path of same length. Let $f_b(1), ..., f_b(I-1), f_a(I)$ be this path. In that case, the system for LSP a can be viewed as a concatenation of two sub-systems, the first one consists of node $(f_a(1), ..., f_a(I-1))$, the second one of node $f_a(I)$. Note, that we assume to de-multiplex all LSPs at the imaginary node $f_a(I+1)$. The map δ_a is defined by $\delta_a(0) = I-1, \delta_a(1) = I, \delta_a(2) = 0$. The service curves are given by:

$$\beta_a^{\delta_a(0)}(t) = \beta_a^{f_a(I-1)}(t) = \min_{\{1 \le i \le I-1\}} C_{f_a(i)}[t - \sum_{i=1}^{I-1} \frac{L}{C_{f_a(i)}}]^+ \tag{5.60}$$

$$\beta_a^{\delta_a(1)}(t) = (C_{f_a(I)} - \rho_b)[t - \frac{L}{C_{f_a(I)}} - \frac{\sigma_b + \sum_{j=1}^{I-1} \frac{L}{C_{f_b(j)}}\rho_b}{C_{f_a(I)}}]^+ \tag{5.61}$$

Because $\beta_a^{\delta_a(RIN_a+1)} = \nu$, we receive:

$$\beta_{R',T'} = \beta_a^{I-1} \otimes \beta_a^I = \min\{\min_{\{1\leq i\leq I-1\}} C_{f_a(i)}, (C_{f_a(I)} - \rho_b)\} \tag{5.62}$$

$$[t - \sum_{i=1}^{I} \frac{L}{C_{f_a(i)}} - \frac{\sigma_b + \sum_{j=1}^{I-1} \frac{L}{C_{f_b(j)}}\rho_b}{C_{f_a(I)}}]^+ \tag{5.63}$$

For the virtual delay, a boundary is given by:

$$d_a(t) = \sum_{i=1}^{I} \frac{L}{C_{f_a(i)}} + \frac{\sigma_b + \sum_{j=1}^{I-1} \frac{L}{C_{f_b(j)}}\rho_b}{C_{f_a(I)}} + \frac{\sigma_a}{\min\{\min_{\{1\leq i\leq I-1\}} C_{f_a(i)}, (C_{f_a(I)} - \rho_b)\}} \tag{5.64}$$

For $C_{f_a(I)} - \rho_b \leq \min_{\{1\leq i\leq I-1\}} C_{f_a(i)}$ this formula can be simplified by:

$$d_a(t) = \sum_{i=1}^{I} \frac{L}{C_{f_a(i)}} + \frac{\sigma_b + \sum_{j=1}^{I-1} \frac{L}{C_{f_b(i)}}\rho_b}{C_{f_a(I)}} + \frac{\sigma_a}{C_{f_a(I)} - \rho_b} \tag{5.65}$$

To compare the two boundaries we simplify our result by assuming the all nodes serve packets at a constant rate C:

$$d_a(t) - d_a^*(t) = \sum_{i=1}^{I} \frac{L}{C} + \frac{\sigma_b + \sum_{j=1}^{I-1} \frac{L}{C}\rho_b}{C} + \frac{\sigma_a}{C - \rho_b} - \sum_{i=1}^{I} \frac{L}{C} - \frac{\sigma_b}{C} - \frac{\sigma_a}{C - \rho_b} \tag{5.66}$$

$$= \frac{\sum_{j=1}^{I-1} \frac{L}{C}\rho_b}{C} \tag{5.67}$$

$$= \frac{(I-1)L\rho_b}{C^2} \tag{5.68}$$

For the valid assumption that $I > 1$ and $L > 0$, this term is positive. The presented formalism thus indicates for networks with a unique link capacity that LSPs should be merged early.

5.4 Grid Resource Management Integration

This section describes the proposed integration of the bandwidth broker into a common Grid resource management framework.

5.4.1 User-Interface

Services offered by a Grid-enabled bandwidth broker should be accessible by the same mechanisms used to reserve and allocate further resources. Hence, a common advance reservation interface is required. Of course, the interface must be designed to transport various specific information. While some attributes are common for all reservations, others depend on the addressed resource. Examples here are the start time attribute which is a generic attribute for all reservations, and the bandwidth attribute which is typically associated with network reservations.

In a Grid environment, service requests are often formulated by using some kind of resource specification language (RSL). This RSL can be used to unify the interface by encapsulating resource specific parameters into the resource specification language string.

Claiming a reservation is always resource-specific, i.e. it does not require the provision of non-resource specific attributes. This is because the exact run-time parameters that need to be specified are always resource specific. In addition, reservation claiming may even be implementation specific. For example, a network reservation may require both ends of the flow to claim the reservation while a job reservation only needs to be claimed once. Though the claiming process by itself is resource specific, a standardized API can be used for many resources. The generic characteristic of this API would again be specialized by an RSL string which defines the resource specific parameters.

An important aspect in the context of network reservation is the fact that these type of reservations are associated with a direction. Any network reservation is uni-directional, from the source downstream to the sink. Service guarantees for TCP based applications, however, also rely on the receipt of acknowledgments. This leads us to propose a multi-layered user interface which consists of a minimum of three layers:

- A resource specific interface, i.e. the external signaling interface of the bandwidth broker.
- A uniform reservation and allocation interface for a single resource.
- A high-level interface which supports the coordinated reservation of multiple resources. This interface is used to establish bidirectional network reservations.

Enabling the specification of authorization and policy information in the API is important for implementing high-level functionalities such as a superscheduler. An example of policy information set by such an entity would be the distinguished name of the owner of a reservation, a list of group memberships, and a list of users or groups allowed to use the reservation in subsequent calls. To ensure the integrity of the policy field, it must be signed by a trusted entity.

In Section 6.1.1 we will present a prototype implementation of a Grid enabled bandwidth broker which is accessible by a unique, resource independent application programming interface.

5.4.2 Inter-domain Signaling and Authorization

This subsection evolves an innovative co-reservation architecture that addresses the issues of inter-domain signaling and authorization. This architecture incorporates two principal elements:

- An *inter-domain signaling protocol* supports the communication of reservation requests and associated authentication and authorization information between resource managers in different domains.
- The abstraction of a core tunnel allows an entity to request an aggregate end-to-end reservation. Users authorized to use a particular core tunnel can then request portions of this aggregate bandwidth by contacting just the two end domains. The intermediate domains do not need to be contacted as long the total bandwidth remains less than the size of the tunnel.

Co-Reservations and Inter-Bandwidth-Broker Signaling

It is unlikely that a single bandwidth broker will control more than one domain, because each administrative domain wishes to have control over the resources it owns. A network reservation for traffic traversing multiple domains must therefore obtain multiple network reservations, as shown in Figure 5.1. Here, Alice wants to make a network reservation from her computer in *source domain* A to Charlie's computer in *destination domain* C. Somehow she needs to contact and negotiate a reservation with BB_A and BB_C as well as the *intermediate domain*, BB_B. There are two approaches to making this happen.

Approach 1: Source-Domain-Based Signaling

Alice, or an agent working on her behalf, can contact each bandwidth broker individually (Figure 5.6). A positive response from every bandwidth broker indicates that Alice has

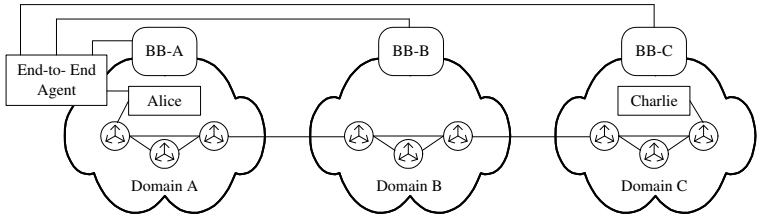

Figure 5.6: Source-domain-based signaling is controlled by a source domain entity, i.e. the End-to-End agent, that contacts all related bandwidth brokers directly.

an end-to-end reservation. However, there are two serious flaws with this methodology. First, it is difficult to scale since each bandwidth broker must know about (and be able to authenticate) Alice in order to perform authorization. Furthermore, if another user, Bob, makes an incomplete reservation, either maliciously or accidentally, he can interfere with Alice's reservation. Such a mis-reservation is illustrated in Figure 5.7.

One possible solution to adopt the approach of multi-domain reservation is to use the API that allows users and applications to manipulate reservations of different resources in uniform ways for each individual network domain. Building an end-to-end network library that facilitates end-to-end reservation for its users could improve this model of adopting the co-reservation approach of multiple resources to the multi-domain network problem. Receiving source and end-point of a network reservation, the library determines the relevant bandwidth brokers and propagates the request to each of them until it reaches the end-domain. The implementation of this API should guarantee that all necessary domains are contacted, but of course there is nothing to stop a malicious user from modifying our implementation to skip a domain. Furthermore, Alice still has to be known by all related bandwidth brokers.

The STARS system [71] adopts a variant of this approach, in which a separate source domain entity—the reservation coordinator (RC)—performs the end-to-end reservation. This strategy alleviates the problems noted above, in two respects: first, in many situations it may be feasible for the RC to be "trusted" to make all necessary reservations; second, all bandwidth-brokers need not be aware of all end-users. However, we still require a direct trust relationship between all intermediate and possible end-domains.

Approach 2: Hop-by-Hop-Based Signaling

The problems just noted are a motivation for the specification of an alternative approach, in which reservation requests are propagated between bandwidth brokers rather than all originating at the end domain. As shown in Figure 5.8, this means that Alice only contacts BB_A, which then propagates the reservation request to BB_B *only if* the reservation was accepted by BB_A. Similarly, BB_B contacts BB_C. With this solution, each bandwidth broker

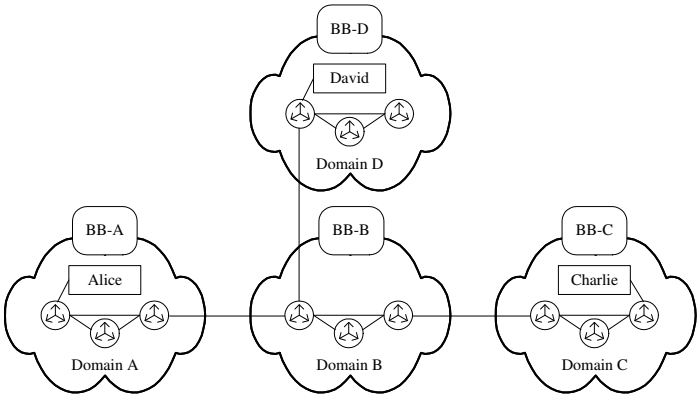

Figure 5.7: Illustration of a consistency problem of source-domain-based signaling. David, a malicious user in domain D, makes a reservation in domains D and B, but fails to make a reservation in domain C, even though he will be sending his marked packets to Charlie in domain C. Domain C polices traffic based on traffic aggregates, not on individual users, so it cannot tell the difference between David's traffic and Alice's reserved traffic. Therefore, there will be more reserved traffic entering domain C than domain C expects, causing it to discard or downgrade the extra traffic, thereby affecting Alice's reservation.

only needs to know about its neighboring bandwidth brokers, and all bandwidth brokers are always contacted. In addition to the hop-by-hop based signaling approach, Figure 5.8 also demonstrates the use of the generic advance reservation API (refer to Figure 6.1.1) to couple a multi-domain network reservation with a CPU reservation in domain C.

With the hop-by-hop based signaling approach, the external signaling interface of a bandwidth broker interfaces not only with an interface procedure of the Grid resource management which maps requests to the particular notation of the external signaling interface, but also with peered bandwidth brokers. This results in a heterogeneous demand for the workflow of the external signaling interface. The discussed state model of the Internet2 community [23] proposes a long term TCP connection to establish a stateful communication between peered bandwidth broker. On the other hand, a Reservation Actuator is accompanying reservations during their lifetime. Hence, there is no need for a long term connection for individual requests. The abstraction of a traffic trunk is the resolution for these heterogeneous demands. While a traffic trunk represents a single reservation for end-domains, it represents the pieces of interest for transient domains: core tunnels. An reservation actuator is accompanying a core tunnel during its lifetime. It subscribes to events signaled by the peered domains and in doing so, it enforces a TCP connection for all entities which have registered a callback which lifetime is related to the lifetime of the core tunnel. For static SLAs, the proposed model is thus conforming to the SIBBS model, as a static SLA is represented by a set of long term core tunnels.

Note that source-domain-based signaling may be faster than hop-by-hop based signaling, because the reservations for each domain can be made in parallel. Also note that the hop-by-hop based signaling should not get starved. It is thus recommended to place the band-

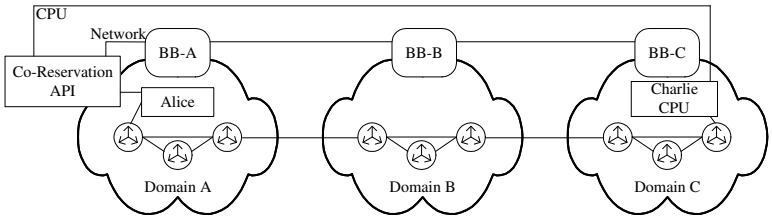

Figure 5.8: Hop-by-hop-based signaling of QoS requests is done using an authenticated channel between peered bandwidth brokers along the downstream path to the destination.

width broker to bandwidth broker communication in the Premium service, or, at least, in a GR service.

Request Parameters

The SIBBS [23] document describes a set of messages and their request parameters used during the communication between peered bandwidth brokers. This dissertation extends the Resource Allocation Request (RAR) and Answer (RAA) of the SIBBS document by two specific parameters: a duration parameter and an object which facilitates an efficient classification of the traffic related to a particular core tunnel. The latter is also introduces to support a signaling of Per-Domain Behaviors. In this context, the proposed extension follows the concept of the RSVP messages, in which the original traffic constraint is kept unchanged, but an additional object is used by each hop to signal its particular capabilities. By iteratively applying service curves on a per-domain base, the concatenation of all service curves can be used as a base for describing the achieved end-to-end behavior. Table 5.1 lists the proposed message fields of a Resource Allocation Request.

The requested network characteristics are specified by the Service Specification object. It regulates both, the arrival curve of the injected traffic and the required service parameters. The injected traffic is characterized by a sustained data rate and by bursts parameters. Optionally, it might also list a short term burst as proposed by the Guaranteed Service of the Integrated Services model (refer to Figure 3.7). Service parameters are the available minimum bandwidth. Optionally it denotes boundaries for delay, jitter, maximum drop probabilities (depending on the color of the marked packet), and costs. Additionally, it contains extended parameters such as deadline file staging (in which the end time attribute is mandatory).

The introduction of the Offered Service object addresses another important issue. Whenever traffic engineering mechanisms are incorporated into the bandwidth broker of a transient domain, the question arises about its instantiation at the edge router. If the allocation of a core tunnel is associated with the placement of an LSP, traffic designated to use the

Protocol Version	The protocol version of the signaling protocol
Request identification	A unique request identification generated by the bandwidth broker of the source domain
Sender identification	Distinguished name of the DS domain that sent the request
Source prefix	IP address prefix for source terminus of the service request. Wild-cards are allowed in the sense of specifying a source domain when the core tunnel flag is set (see below)
Destination prefix	IP address prefix for destination terminus of the service request. Wild-cards are allowed in the sense of specifying a destination domain when the core tunnel flag is set (see below)
Ingress router address	IP address of the interface between two domains for which the sending domain is requesting service
Start time	Time at which the service should be available first
Duration	Specific time the service request has to be applied
End time	Optional parameter which allows a variable placement of the reservation with the interval $[start\ time, end\ time]$
Flags	Two flags are currently proposed: A probe request is used to support the operation of a super-scheduler and a core tunnel request is used to support aggregated reservations
Service Specification	Requested service parameters
Offered Service	The combined Per-Domain Behaviors applied by all upstream domains. Note that this object is updated by each bandwidth broker. It contains the proposed DSCP marking
Core tunnel voucher	Credential used for authorizing the use of a core tunnel

Table 5.1: Resource allocation request message format. The Offered Service object is introduced to facilitate an efficient classification of the traffic of a particular core tunnel.

core tunnel must somehow be mapped to the related LSP. For efficiency reasons, the edge router should not have to do the full classification of an edge router of a DS source domain, i.e. to process the full IP header and higher-level headers. Instead, the "Offered Service" object is used to signal a particular DSCP for the traffic of the core tunnel. Note that the particular DSCP does not necessarily indicate a new per-hop behavior. Instead, it is mapped to one of the existing PHBs, namely the EF PHB, or, if existent, the AF PHB. Core routers of each domain define reasonable sized classes of DSCPs which members are all treated equally. The DSCP is thus used as a notation for a core tunnel. This concept follows the proposed pragmatic deployment principle which does not rely on many of those DSCP classes. In fact, if appropriate from a billing perspective, even a single DSCP class can be used to serve Grid applications with the required service. Omitting redundant information in the Resource Allocation Answer (RAA), the message fields of the RAA are listed in Table 5.2:

Note that it is not the intention of this dissertation to present a new standard for an inter-domain bandwidth broker protocol. Instead, by following the "easy-to-deploy paradigm" the design extends an existing proposal which is under consideration of the related community.

Protocol Version	The protocol version of the signaling protocol
Request identification	A unique request identification the response is related to
Sender identification	Distinguished name of the DS domain that sent the answer
Ingress router address	IP address of the interface of the router which domain is sending the response
Start time	Time at which the service should be available first
Duration	Specific time the service request has to be applied
Offered Service	The combined Per-Domain Behaviors applied by all downstream domains. Note that this object is updated by each bandwidth broker. It contains the required DSCP marking of the peered upstream domain.
Core tunnel voucher	Credential used for authorizing the use of a core tunnel

Table 5.2: Resource allocation answer message format. The Offered Service object lists the marking the packet marker of the upstream domain should apply for the core tunnel.

Policy Information

A complicating factor in a multi-domain environment is that different domains may wish to enforce different policies concerning who can use their resources. For example, in Figure 5.9, the three bandwidth brokers specify three different policies:

- The source domain bandwidth broker, BB_A, specifies that Alice is allowed to use as much bandwidth as she wants, up to the maximum available, except during business hours when she is restricted to 10 Mb/s.
- The intermediate domain bandwidth broker, BB_B, specifies that up to 10 Mb/s can be allocated to anyone who is a member of group "ATLAS experiment" or who can provide a capability provided by community "ESnet."
- The destination domain bandwidth broker, BB_C, specifies that it will only accept reservations above 5 Mb/s, if the requester can provide a capability provided by community "ESnet" and if she can present a valid reservation for a computing resource in domain C.

In general, a bandwidth broker making a decision must be able to consider:

- request parameters, e.g., the destination domain and the amount of bandwidth required,
- authentication information, e.g., a public domain credential for the originating user,
- authorization information such as assertions regarding group memberships (perhaps originating from the user or the source domain) and cryptographically signed capabilities issued by various authorities,
- SLA information such as traffic engineering parameters.

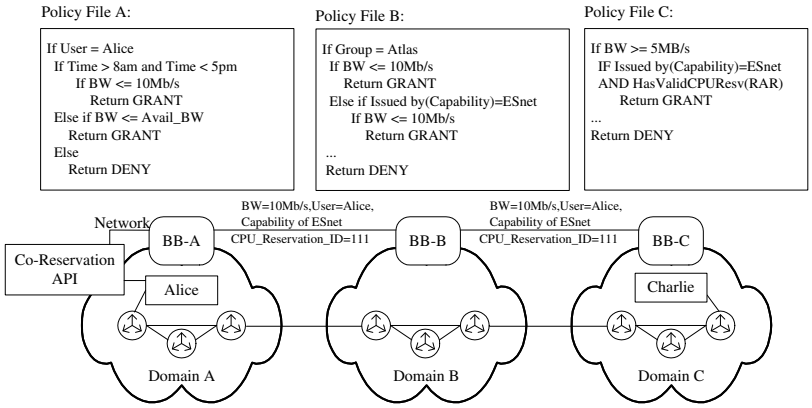

Figure 5.9: A multi-domain environment where each bandwidth broker enforces a specific list of reservation policies. The user Alice is making a network reservation request, referring to an existing CPU-reservation in domain C. Each bandwidth broker will evaluate each request with respect to its local policy file.

Let us assume that either the end user or the bandwidth broker of the source domain acting on behalf of the end user contacts a policy server such as an Akenti [125] server. This policy server provides the related policy information based on the request, its use conditions, and the identity of the requester. This policy information is propagated along with the user's request. However, the propagation protocol should not make strong assumptions on the actual syntax of this policy information. It should handle simple attribute-value pairs which might be signed by the assigning entity as well as capability certificates.

By separating authentication and authorization issues one can facilitate the flexible propagation of different policy related information. As long as the protocol ensures that the end-entity can approve the integrity and the authenticity of the received information, authorization decisions can be made without depending on specific features of the language expressing the policy attributes. Therefore, the same propagation protocol can be used for different policy representations.

The important aspect here is

- the protocol is independent of policy syntax, and
- the different domains need to agree on the syntax, and
- the syntax in the figures is therefore just an example.

From that perspective, the actual syntax of the use conditions and capabilities [125] described as policy file in Figure 5.9 and the illustrated format of the policy information represents one example scenario of the propagation protocol.

Notice that an implication of this discussion is that the evolved architecture must provide mechanisms for communicating information securely between bandwidth brokers.

Architecture Overview

We can now proceed to define the proposed architecture. Let us assume a set of bandwidth brokers that communicate via an inter-bandwidth-broker signaling protocol. A source bandwidth broker accepts incoming requests that contain the information listed in the preceding section, such as request parameters, authentication information, and authorization information (assertions and/or capabilities).

We introduce an entity called a policy server that encapsulates a bandwidth broker's admission control procedures. When a request comes in, it is forwarded to the policy server which executes local policy and passes back a result ("yes" or "no") and a modified request. Though an implementation of a policy server is not within the scope of this dissertation, the general ability to express diverse authorization policies is an import issue. The evolve framework must therefore be capable to handle the following authorization mechanisms:

- Authority based on validated *assertions* concerning group membership. In this case, the policy might say "approved if group server P validates the user as a physicist"; if the user's request includes the assertion "I am a physicist", then the policy server verifies that assertion by contacting that group server, passing the user's supplied identity certificate. The group server then verifies whether the user is a member of the group and responds appropriately.

- Authority based on cryptographically signed *capabilities* issued by various authorities [94]. In this case, the policy might say "approved if the user supplies a capability of type C issued by authority A;" if the capability C is supplied, then the policy server verifies its validity and responds appropriately. One representation of capabilities is to encode the capability attributes in the extension field of an ITU X.509v3 certificate [74], issued by a specific Community Authorization Server (CAS) [102] being developed within the Globus project.

- Traditional *access control lists* may also be of interest, expressed in terms of the identities of individuals who are allowed to use resources.

A Transitive Trust Model for Signaling Policy Information

Having defined the principal elements of the proposed architecture, we now describe the inter-bandwidth-broker signaling protocol in detail and explain how (a) it ensures secure transmission of information between bandwidth brokers and (b) establishes direct trust relationships between end domains, as required for the establishment of tunnels. The actions that are performed are different for the source domain, for intermediate domains, and for the destination domain.

Source Domain

Service requests are originated by a user (or agent acting on their behalf) which signals the demand to the bandwidth broker of the related administrative network domain. In addition to the basic bandwidth request, such as 10 Mb/s of guaranteed bandwidth, this request may include additional information such as a cost that the user is willing to accept and assertions and capabilities as described above. When such a user request arrives, the source domain bandwidth broker performs four steps:

- The bandwidth broker contacts the policy server to verify that the user-provided information is correct, and that the user is authorized to make the request in the local domain.
- The bandwidth broker receives additional domain-wide information from the policy server. This information is used to identify additional constraints that might have to be added to the reservation request. This may include groups in which the end-domain resource requires membership, additional cost offers for the particular request, any information relevant for traffic engineering purposes for downstream domains, or specific requirements derived from the contract with the peered domain, such as parameters for treatment of excess traffic or reliability parameters expected for this service [128].
- The bandwidth broker decides whether or not the request can be satisfied within the local domain, based both on the traffic profile and the policy constraints.
- If the reservation request can be granted locally, the bandwidth broker forwards the request to the next bandwidth broker in the network path, along with any additional information that was added. This additional information facilitates a signaling path tracing as well as the propagation of identity information. This allows the establishment of a direct mutually authenticated channel between source- and end-domain when the tunnel is actually used.

Intermediate Domain

Whenever an intermediate bandwidth broker receives a message from the upstream bandwidth broker, it checks whether the requested traffic profile conforms to the related SLA, and, if this is the case, it may add additional information such as capabilities and policies, and will forward the request downstream. It may use SLS-related information added by any upstream domain, if they exist and are relevant for this decision.

Whenever a request is denied by one domain, the event is propagated upstream to inform the user of the reason for the denial.

Destination Domain

The bandwidth broker of the ultimate end domain makes the final authorization decision based on its local policies, using (as in the case of the intermediate domains) any or all relevant information supplied in the request, whether request parameters, identify certificate, assertions, or capabilities.

The Signaling Protocol

We now look more closely at the security issues associated with the proposed inter-bandwidth-broker signaling protocol. There are two issues:

- Messages between bandwidth brokers should be mutually authenticated
- Because trust is not transitive in general, the protocol must accomplish a trustworthy model for transporting the policy and additional information end to end.

The direct signaling between peer bandwidth brokers used in the above description can easily be secured using Secure Socket Layer (SSL) or Transport Layer Security (TLS) [32]. While SLAs are used to regulate the services between two domains, the evolved model extends this agreement by adding information to facilitate the trust relationship between two peered bandwidth brokers. This information includes the certificates of the peered bandwidth brokers as well as the certificate of the issuing certificate authority, all used during the SSL handshake.

A common way to ensure the integrity and authenticity of messages is to use digital signatures. In the particular case, this works as follows. A user requesting a service augments the request with any relevant additional information, such as a supplied reservation handle, and signs the resulting augmented request with her private key before it gets propagated. The source domain's bandwidth broker might further augment the request—such as information received from a policy server—and sign the resulting larger request with its own private key. A complete request therefore is comprised of a collection of information, each signed by the entity that added it. The signatures both assert the authenticity of the information and allows for the tracking the path taken by a request as it moves from bandwidth broker to bandwidth broker.

When a request is received at the destination domain, the bandwidth broker checks local policy and resource availability. If these checks succeed, then the bandwidth broker adds its own signed policy information and propagates the modified request to the previous intermediate domain bandwidth broker, again using SSL/TLS. The approval therefore propagates back to the source domain, with each intermediate domain referring to local SLA and SLS information as it verifies that it can approve the request.

Establishment of core tunnels is supported by a resource allocation request (RAR), which is the dynamic establishment of a direct signaling channel between source- and end-domains. Because of this direct connection, it must be possible for the end-domain to derive the identity of the source domain's bandwidth broker.

One technical problem raised by this approach is access to public keys. The approval of a digital signature requires access to the public key of the signing subject. This access can be accomplished by one of the following techniques:

- Distribute all relevant certificates within all requests. Supposing that the issuing authority is known and trusted, one can check the authenticity of the signature. However, there is a question of whether there is a way to facilitate the approval of a signature of entities without a direct trust relationship and in the absence of cross-signed CAs. We address this problem by having each domain add the certificate of the upstream domain—known because of the SSL handshake—and sign it. This web of trust allows each domain to access a list of key introducer [58] when deciding whether to accept the public key stored in the certificate.

- Maintain a certificate repository accessible through secure Lightweight Directory Access Protocol (LDAP).

  Upon receipt of the reservation specification, C would extract the distinguished name (DN) of A from it, and would search in the certificate repository for the related public key. It is important to note that there has to be a strong trust relationship with the repository.

- Completely decouple the distribution of policy information from bandwidth broker-to-bandwidth broker communication, i.e., transport it out of band.

- (Restricted) delegation mechanisms could be used to propagate authorization attributes, by having each bandwidth broker impersonate the caller's identity.

While each of these solutions has interesting characteristics, the first solution is to be preferred because it offers a flexible framework for trust decisions supporting different security levels. The following notation is used to describe the proposed mechanism and its advantages:

- *res_spec* reservation specification of the user

- *Capability_Cert* denotes authorization information in any valid representation. The information is typically signed by an issuer, i.e. a policy or an authorization server. Examples are Attribute Certificates [39], capabilities [94], or Impersonation Certificates [129] containing authorization attributes in its extensions. In this section, the term $Capability\_Cert'_A$ is used to indicate that entity A has issued a capability. A detailed description of this procedure can be found at the end of this section. Note

that the delegation is only performed when capabilities are transported. For other representations this field might be empty.

- *pkey*$_A$ private key of entity A
- *cert*$_A$ X509 certificate of entity A
- *sign*$_{pkey_A}$*(attributes)* adds a signature to the given attribute list using the private key of entity A
- *DN*$_A$ distinguished name of entity A

We assume that the bandwidth broker in domain A receives the following information from User U:

$$RAR_U = sign_{pkey_U}(\{res\_spec, DN_{BB_A}, Capability\_Cert'_{CAS}, Capability\_Cert'_U\})$$

Because RAR_U was received through a mutually authenticated channel, it is valid to assume that the bandwidth broker in domain A has access to the user's certificate. This information facilitates the approval of the received capability certificates, which were issued by some authorization servers, because the granted capabilities were passed to BB_A using the user's private/public key pair as proxy key. Once the request was approved, it is extended with the user's certificate and the DN of the downstream bandwidth broker, as well as with additional policy information, if necessary, and signs the new message using its private key:

$$RAR_A = sign_{pkey_{BB_A}}(\{RAR_U, cert_U, DN_{BB_B}, Capability\_Cert'_A\})$$

When BB_B receives this, it adds BB_A's certificate and the distinguished name of the downstream bandwidth broker to RAR_A. If necessary, it will add additional policies and capabilities, signs the whole message, and transmits it to C:

$$RAR_B = sign_{pkey_{BB_B}}(\{RAR_A, cert_A, DN_{BB_C}, Capability\_Cert'_B\})$$

Note that BB_C is able to check the signature of RAR_B because it does have access to the certificate of BB_B exchanged during the SSL handshake. Additionally, BB_B introduces

the public key of BB_A by transmitting its certificate. BB_A however, as source of the request, did approve the SLA with domain B by listing the DN of BB_B in its request. BB_C can now decide whether it trusts BB_B's introduction.

Now let us assume that RAR_N specifies the message submitted by the n-th bandwidth broker of the path between source and end-domain. Furthermore, let us assume that the n+1-th bandwidth broker is not the one of the end-domain. Then we can describe the message created by the n+1-th bandwidth broker as:

$$RAR_{N+1} = sign_{pkey_{BB_{N+1}}}(\{RAR_N, cert_N, DN_{BB_{N+2}}, Capability\_Cert'_{N+1}\})$$

While the proposed protocol permits direct access to the transported information whenever appropriate, it also makes it possible to check signatures without a direct trust relationship. For example, in the case above, let us actually resolve what BB_C would receive:

$$\begin{aligned} sign_{pkey_{BB_B}}(\{sign_{pkey_{BB_A}}(& \\ & \{sign_{pkey_U}(\{res\_spec, DN_{BB_A}, \\ & Capability\_Cert'_{CAS}, Capability\_Cert'_U\}), \\ & cert_U, DN_{BB_B}, Capability\_Cert'_A\}), \\ & cert_A, DN_{BB_C}, Capability\_Cert'_B\}) \end{aligned}$$

BB_B's certificate is approved by the SLA and the SSL handshake. BB_C can therefore be sure that the bandwidth broker of domain B has approved the receipt of the message

$$\begin{aligned} sign_{pkey_{BB_A}}(\{sign_{pkey_U}(\{res\_spec, DN_{BB_A}, & \\ Capability\_Cert'_{CAS}, Capability\_Cert'_U\}), & \\ cert_U, DN_{BB_B}, Capability\_Cert'_A\}) & \end{aligned}$$

from a trusted entity presenting certificate cert_A. Note that C also knows that B has established this trust relationship based on a contract, i.e., B has signed a contract that enforces it to trust the subject capable of using cert_A. Depending on the level of trust C is actually requiring, this permits a trust relationship to A, as B was

- approving that BB_A was able to use the private key corresponding to cert$_A$
- has a trust relationship to A based on a contract
- has a known trust relationship to C based on a contract

Checking its own security policy which might limit the depth of an acceptable trust chain, BB_C may accept the public key of cert$_A$, and use this to approve the received information.

The proposed model offers a flexible and realistic solution for propagating policy information end to end. It is flexible because it does not enforce a specific security policy: instead, it offers access to all relevant information. It is realistic because it follows existing trust relations. From an accounting perspective there is already an accepted transitive billing scheme. Whenever a domain actually bills the requesting entity for the use of the network service, SLAs are already used to set up a transitive billing relation in multi-domain networks. When network traffic enters domain C through domain B, it is billed using the agreement between B and C. B as a transient domain, however, would also bill traffic originating from a different domain using the related SLA. Finally, the source domain would bill the traffic against the originator.

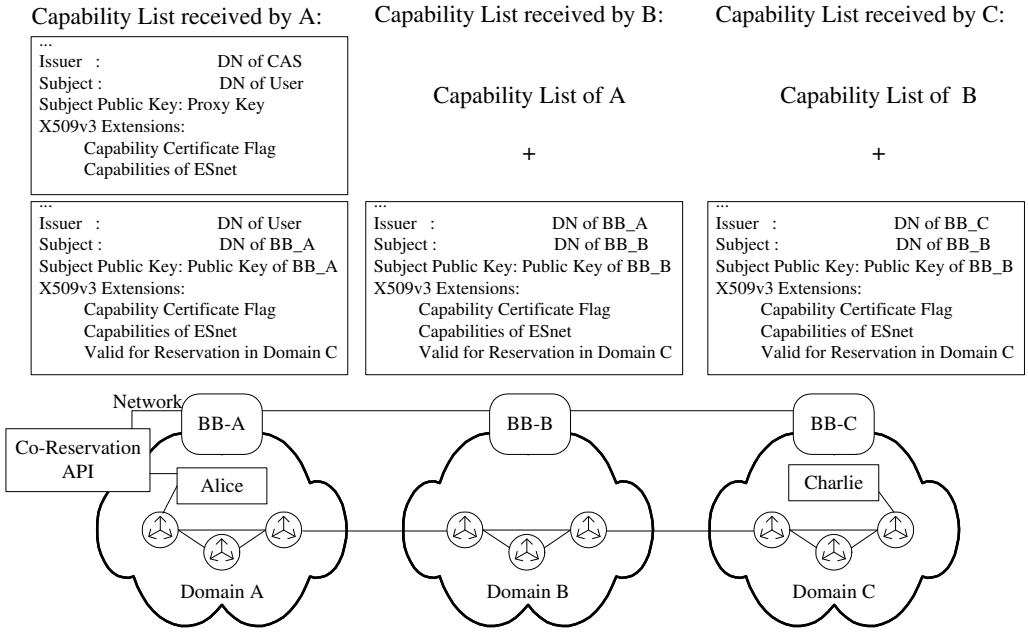

Figure 5.10: Capability certificates received by each bandwidth broker during the proposed end-to-end signaling process.

Propagation of Capability Certificates

The model also supports capabilities issued by community authorization services via mechanisms that allow the end-domain to use a granted capability for authorization purposes. Instead of using the private key corresponding to the public key listed in the capability, the bandwidth broker of the end-domain will use its own private key, together with the full chain of messages. This chaining can be accomplished by following the cascaded authorization mechanism proposed by Neuman [94]. In his model each subordinate server signs the received capabilities using the private key of the corresponding public key stored in the capability. Neuman used proxy-key pairs to fulfill this task. In the proposed model, the bandwidth broker of the source domain uses the public key of the peered downstream domain as public proxy key.

To describe the proposed protocol more precisely, we construct a use scenario (Figure 5.10) where the user has received a capability certificate by some Community Authorization Server (CAS) [102] during the "Grid-login" process. Let us assume that the capability certificate simply contains all capabilities of the ESnet group in the X509v3 extension field. The certificate itself lists a public proxy key, the DN of the user (potentially modified to indicate that this is a capability certificate) and the CAS, as well as the signature of the CAS. In addition to the capability certificate, the user owns the private key corresponding to the public proxy key. Whenever a service is requested, the related server receives the capability certificate and requests prove of the knowledge of the private proxy key. This step, however, can be viewed as authentication. Whenever the authenticity of the capability certificate is approved, a policy engine can directly use its attributes, such as the group membership, to decide whether the request can be granted or not.

In the constructed scenario the user now requests a network reservation from a host in domain A to a virtual reality device in domain C. To describe the delegation process, we introduced the generic notation Capability_Cert′. Here, we clarify the implementation of this notation for capability certificates. To delegate the capability cert to BB_A, the user creates a new capability certificate. The subject of this new certificate is BB_A. Instead of creating a new public key, the SSL handshake of the protocol allows to insert BB_A actual public key to this certificate. Finally, the extensions of the original capability certificate are copied, i.e. the group membership, extended by an additional restriction "valid for RAR". Instead of signing the new certificate with the user's private key, it is signed by using the private proxy key.

BB_A now receives two capability certificates. The original one issued by CAS and the one issued by the user. Note that BB_A can prove that it actually posses the new capability certificate by proving the knowledge of the related private key: $pkey_{BB_A}$. Note that the remaining fields of RAR_U are not needed in this context. Their purpose is to implement the introductory model which facilitates the establishment of a tunnel between source and end-domain.

Now BB_A delegates the received capabilities to BB_B by creating a new certificate. Therefore BB_B receives three capability certificates. One issued by the CAS, one by the user, and one by BB_A. Finally, BB_B delegates this to BB_C which posses four capability certificates.

To authorize the request, BB_C can now submit the certificate chain to a policy engine which:

- checks that CAS was issuing a capability certificate for the user,
- checks that the user was able to use the private proxy key during delegation to BB_A,
- checks that BB_A delegated the capability to BB_B, because the new certificate was signed using $pkey_{BB_A}$,
- checks that BB_B delegated the capability to BB_B, because the new certificate was signed using $pkey_{BB_B}$,
- checks that BB_C actually owns the capability certificate by requesting a prove of the knowledge of $pkey_{BB_C}$,
- checks that the validity of all capabilities, i.e. whether some entity did change them inappropriately during delegation,
- uses the ESnet capabilities for authorization purposes.

This leads to a consistent delegation model for policy information which is a prerequisite to embed the network as a Grid resource which is managed by the resource manager of the source domain.

5.5 Handling of Grid Applications

To illustrate the use of the framework, this section describes examples of the proposed handling for each of the classes of Grid applications.

5.5.1 Distributed Supercomputing

The differentiation between computation and communication phases results in a bursty traffic profile of distributed supercomputing applications. Whenever the communication libraries have to pass the TCP protocol stack to reliably communicate with remote peers, the propagation delay of the messages is influenced by the behavior of TCP. In this context, the impact of lost packets due to network congestion is a potential problem.

To describe this impact, we have to distinguish between two scenarios:

- The available window size of the sender is larger than the message size, i.e. both peers are capable of buffering a full message. This is a likely situation for applications which are using large socket buffer sizes. When this scenario is coupled with a low write frequency, TCP can pass the data to the IP layer without awaiting the acknowledgments. If no downstream packet drops occur, the TCP implementation of the receiver delivers the data to the application at the time, the last segment was received. Hence, the delay is more or less independent from the properties of the upstream path. In case of congestion, where downstream packets get dropped, the receiver signals this event to the sender by either duplicate or selective acknowledgments (SACK). Whenever SACK is not supported, or not able to solve the retransmission, the application experiences the retransmission of all packets starting from the first packet drop. The delay between transmission and retransmission is thus influenced by the conditions of the upstream path and by the ability of TCP to handle the situation without any timeout. Whenever timeouts occur, the application experiences a dramatical increase in delay. Figure 2.4 illustrates this scenario. Additional problems arise by the fact that TCP responds to packet loss by either falling into its congestion-control or into its slow-start phase. Here, throughput is limited by the congestion window size, a situation which is similar to the second scenario we now describe.

- Whenever the message does not fit into the socket buffer, i.e. the former message is not successfully delivered yet, the transport of data segments relies on the receipt of further acknowledgments. The end-to-end message propagation delay therefore depends on the properties of both, the downstream and the upstream path. Here, packet drops not just influence the transmission rate by the time it takes to identify and recover them, they also cause a reduction of the congestion window size and thus further limit the transport of data segments. Because network reservation are uni-directional, an upstream reservation must be performed even if the communication is uni-directional. A higher-level library is appropriate to automate the reservation for uni-directional TCP connection. Assuming the use of path MTU discovery [122], the ratio between TCP data segments and acknowledgment (ignoring delayed acknowledgments) is $1500/40$.

From the perspective of a distributed supercomputing application this is a dramatical change of behavior expectation compared to a tightly-coupled cluster environment where lower protocol layers typically provide an efficient reliable data transport mechanism. The hardware flow control of the popular Myrinet interconnection technique of Myricom Incorporation, for instance, does not drop packets unless the receiving network interfaces fail to drain the network [105]. We propose to address this problem by a Guaranteed Rate (GR) service which does not drop any conforming packet.

Access to a GR service offered by a bandwidth broker requires the specification of reservation parameters. However, the requester must expect that any service instantiation limits

the amount of allowed bursts. Reserving the peak rate of the communication phases would be a simple resolution of this problem, but also a waste of the scarce and potentially costly resource of the GR service. The ability to trade-off between the acceptable constraint of the injected traffic and the benefit the application achieves by the related GR service is what we propose instead. To facilitate this trade-off, this dissertation suggests the use of flow-based traffic shaping at the source domain. While the proposed GR service offers a tunnel of a particular average rate capacity, the token-bucket is configured to allow some bursts. The intention here is to support TCP-based application by allowing bursts of up to a full TCP window. Traffic shaping is used to assure that these bursts are not injected to the core network.

Following the "easy-to-deploy" paradigm, this solution does not rely on the ability of the end-host to shape out the traffic appropriately. Instead, it relies on standard building blocks of IP-based QoS and assumes that— in principle — the Control Procedures of the bandwidth broker control the shaping function of the ingress routers to shape out traffic of individual flows. Of course, traffic shaping has to be done without enforcing packet drops due to shortage in queuing space. However, this assumption is not unrealistic as commodity hardware support buffer spaces in the order of MBs [115]. We can therefore assume buffer capabilities for reasonable large window sizes.

To apply a per flow-based traffic shaping, the policing function of the ingress router is extended by a related traffic shaping configuration on the output interface. While the token bucket used to police incoming packets allows bursts of up to one message, the related traffic shaping of the output link is limiting the actual burst size in the core of the network. The intention of this configuration is to adapt the injected traffic profile to the reserved rate. Assume for example a message size of 1Mb which is traversing to the ingress router at a link speed of 100Mb/s. Now assume that the application reserved a bandwidth of 50Mb/s and the underlying shaping implementation is able to shape the traffic out at a constant bit rate. Whenever the packet transmission does not rely on the receipt of acknowledgments, the end-to-end message propagation delay is given by the delay D of the first segment, plus the delay caused by transmitting the data of a single message at the available rate: 20 milliseconds. The delay of the first segment is a property of the transition of all Per-Domain Behaviors. If implemented appropriately, the bandwidth broker of the source domain receives this value when it processes the Reservation Allocation Answer and can thus prove whether the requested service was achieved. Based on the ability to control the end-to-end delay of messages of a given size, users are able to balance their distributed supercomputer application efficiently.

From an application point of view, the generic advance reservation interface to Grid resources must be embedded somehow into the MPI standard. A standards-compliant extension of the MPICH implementation of MPI [61] using MPI's attribute mechanisms to communicate with the underlying QoS system is presented in [113]. Hence, the access to a GR service is provided in conformance to the MPI standard.

In addition to the intra-application related network demand, distributed supercomputer applications also rely on the availability of the required resources. This includes the access to the required data. Deadline file staging is thus an important requirement for any scheduling service which is responsible for selecting the resource candidates of a distributed supercomputer application. This specific request is discussed in the following subsection.

5.5.2 High-throughput Computing

High-throughput applications raise two specific service demands:

- The need for TCP-based deadline file staging
- The need for efficient transport strategies to support an efficient remote checkpointing

The demand for deadline file staging is addressed by reserving the required capacity at a GR service. To ensure the deadline conforming use of the dedicated capacity, advanced pacing mechanisms are required, where either the application is able to adapt its transmission rate to the available capacity, or the bandwidth broker itself controls the use of per-flow based traffic shaping with the intention to pace TCP traffic by queuing a full TCP window.

By additionally applying for unused GR capacity the model improves the economic use of the network. The underlying idea is to actively reserve the amount of bandwidth which is required to meet the deadline. This reservation is non preemptive. If the Control Procedure of the bandwidth broker recognizes, however, that there is more guaranteed bandwidth available than requested, it assigns an additional preemptive reservation to the requester. Using pacing mechanisms, the file transfer can now actively use this oversized GR tunnel. Whenever a further reservation request is issued, the file transfer has already succeeded in transferring an additional amount of data than it was formerly required to meet the deadline. This knowledge can then be used to reduce the required minimum bandwidth and it thus available to other service requests.

The economic use of the network can be additionally improved by also applying for unused best-effort bandwidth. The fundamental idea is to split the data set into chunks, which are transported by two file transfers: one is assuring the deadline and is applying for the GR service and one is using an highly-tuned transfer. In order to allow for fairness among competing and responsive BE flows this dissertation proposes to map those bulk transfers to a less-than BE service. Within the Internet2 QBone-initiative the DS Scavenger service [123] was introduced to provide such a less-than best-effort service for high-bandwidth and also unresponsive flows. By applying the Scavenger service the bulk transfer application not only collects unused GR bandwidth but also unused best-effort capacity at the price that

the Scavenger service can be starved by best-effort traffic. Whenever the bulk transfer application succeeds in transmitting an appreciable amount of additional data by applying the Scavenger service, forward signaling of a reduced guaranteed bandwidth requirement to the reservation manager is applied to decrease the reserved capacity and thus reduce costs and allow for a smaller blocking rate of the reservation manager, due to a higher available capacity.

Similarly to the support of deadline file staging, the provision of an efficient transport strategy for remote checkpointing is based on access to GR services together with the ability to control the use of such channels, i.e. to avoid packet loss due to exceeding the size of the GR reservation.

5.5.3 On-demand Computing

The dynamic nature of on-demand computing application is addressed by advanced control procedures and feedback mechanisms. It is well-known that programs relying on feedback and adaptation still require a certain minimum performance from the network [5, 106]. Adaptation cannot solve all problems for an application if best-effort service does not provide sufficient bandwidth to meet these minimums.

The proposed architecture reflects these concerns by an alternative approach based on reservations and explicit dynamic feedback mechanisms. The support of adaptive applications is discussed in Section 5.6.

5.5.4 Data-intensive Computing

Data-intensive applications do have challenging throughput demands. The development of many tools and services is underway to tackle this problem. Due to the fact that the throughput of TCP flows is strongly influenced by the size of the used socket buffers, socket buffer tuning algorithms and cookbooks are under consideration.

There is, however, a fundamental problem caused by the “easy-to-deploy” paradigm: the semantical gap between socket buffer interface and the protocol capabilities of TCP cannot be closed easily. While the protocol itself introduces the window scale option during the three-way handshake, there is no way in commonly used operating systems to explicitly set this option by issuing a specific setsockopt()-call. There is even no standardized way to influence the value of this important TCP option. In fact, the window scale option is derived from the socket buffer size used during the connect()- and listen()-call. Unfortunately, this selection is done on a minimum base which means that the minimum required window-scale option is used. To explain this mechanism in more detail, suppose that the used socket buffer size would be 50KB, 100KB, and 150KB.

In the first case, the window scale option would be not used at all. Because the TCP protocol does not allow to update the window scale option afterwards, the maximum socket buffer size for this session would be 64KB. Any socket-buffer tuning library recognizing a buffer shortage could not increase the existing buffer space beyond this threshold.

In the second case, most operating systems would select a window scale option of 1. Hence, the maximum socket buffer size would be 128KB. In the final case, the window scale option used is 2 which results in a maximum buffer size of 256KB.

In general, the benefit of dynamic socket buffer tuning algorithms which adjust the buffer sizes after the connection has been initialized is limited by either the old 64KB threshold, when no window scale option was set, or by a factor of 2. The Web100-Project [133] is trying to address this problem, but its success strongly depends on its influence of major operating system vendors.

Following the "easy-to-deploy" paradigm, the proposed framework does not rely on modified versions of the TCP protocol stack. Instead, it assumes that tools will help application to set the socket buffer sizes above the minimum threshold and claim that the socket buffer size and the expected boundary for the round-trip time is propagated to the bandwidth broker. When the reservation is instantiated at the source domain, the bandwidth broker assures that the resulting ingress router configuration supports those buffer sizes by declaring the injected traffic as reservation conforming if it is not exceeding the desired rate plus a burst of the given socket buffer size. However, the ingress router will shape out traffic according to the reserved rate.

Again it is proposed to combine the presented mechanisms with the ability to use the Scavenger service in conjunction with a highly-tuned transfer program such as "gridftp" [2].

5.5.5 Collaborative Computing

Collaborative computing environments often rely on real-time capabilities. A video-conferencing-tool must present audio and video information in a given period of time, or the data is obsolete. Similarly to this delay threshold, a jitter bound is another important aspect for this type of application. In a Differentiated Services environment, this demand is addressed by a Premium service.

A teleimmersion application reacts to user actions by adapting the virtual world. For example, sensors detect the user's location and orientation to enable the simulation program to perform calculations to update what the user sees. This creates a highly dynamic environment, because it is not known in advance what the result of this simulation process is. For example, one result of this calculation could be that a new object created by the users has to be stored in the database. Another example would be a major update of the visualized virtual world when a user walks around an object and changes their view. Any

portion of this process may be distributed: the raw data may be sent to the client's computer to be processed and visualized or processed data may be sent to the client's computer be rendered [29, 49]. Any of this data that is sent may have bandwidth needs that are difficult to predict. A QoS framework should therefore support feedback mechanisms to inform the application when it has made a reservation that is too small or too large.

5.6 Support of Adaptive Applications

So far, Figure 5.2 illustrated the basic building blocks of the proposed bandwidth broker architecture. We now refine this view with respect to the support of adaptive applications.

5.6.1 Refined Architectural Framework

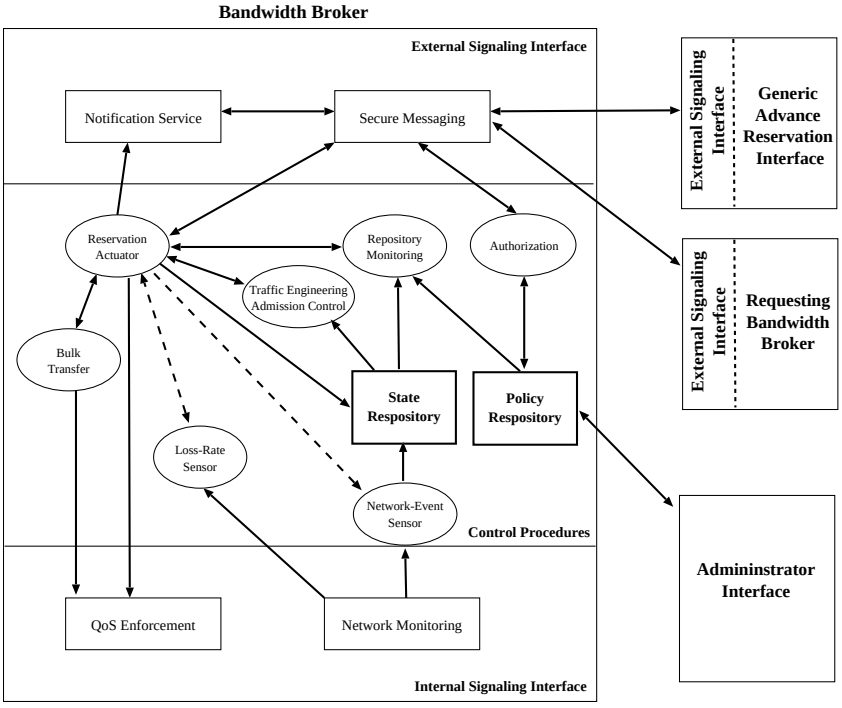

Figure 5.11: Refinement of the proposed architecture to support adaptation. The external signaling interface consists of a Notification Service which propagates asynchronous events to subscribed and authorized subjects and a secure messaging interface which ensures the authenticity of messages. The Control Procedures consist of a reservation actuator, sensors, and decision procedures. Finally, the internal signaling interface is constituted by a QoS propagation mechanism and a supervisor interface.

As illustrated in Figure 5.11, the interaction between bandwidth broker and requester is encapsulated by the internal protocol of the external signaling interface. All messages are

authenticated by a Secure Messaging procedure which uses an Authorization decision procedure to validate whether the requester is allowed to perform the related operation. Note that this check involves the consideration of policy rules specified in the Policy Repository. Once a message has been approved, it is passed to a process which is called the Reservation Actuator. This process is responsible for a particular request during its lifetime. It is initiated when a new request arrives and terminated when the request finishes. Because service requests are represented by traffic trunks (refer to Section 5.1.4), the first task of a newly created Reservation Actuator is to initialize a traffic trunk. While traffic trunks are persistently stored in the State Repository, they are dynamically mapped to the underlying topology. Hence, the Admission Control procedure can incorporate Traffic Engineering mechanisms during its decision process and can thus consider the current state of the controlled domain. Once a traffic trunk was successfully mapped to the topology, the Reservation Actuator subscribes to important repository events such as link failures. A Repository Sensor is used to filter the related events and to prepare their appropriate propagation.

Reservation Actuator, sensors and decision procedures act in concert to achieve adaptive control. For example, a Loss-Rate Sensor is used to monitor the policing function of the edge router for a particular flow or aggregate. Whenever packets get dropped, the sensor propagates the information to the Reservation Actuator which reacts to the event by either updating the reservation automatically, or by invoking the notification service to inform the application about the problem. An asynchronous decision procedure in the associated application can then be executed to determine whether to reduce the sending rate or, alternatively, modify the existing reservation.

The lifetime of reservations starts with the reservation requests and ends with either an explicit cancellation, or simply by the arrival of the end time. As stated above, a particular procedure is accompanying each individual reservation during its lifetime: the Reservation Actuator. All relevant events are handled by this core function.

A requirement for adaptive control is the ability to determine the state of relevant system components and in particular the ability to detect state changes. This capability is provided via Sensors. These processes gather relevant information. The event notification is provided via some form of event service or callback mechanism. Figure 5.11 illustrates the use of two sensors by decision procedures.

The loss-rate sensor monitors the policing function of a particular flow at the edge of the network. Whenever packets are explicitly dropped, this information can serve to propagate the information to the application that it is sending too fast, i.e. that it has an inadequate reservation.

The second proposed sensor is a generalization of the loss-rate sensor: a network-event sensor. It is the basic building block for assuring the consistency between the state repository and the state of the network. The implementation of this sensor varies from periodically

pulling the relevant information out of the edge routers to a passive participation in routing protocols.

The third component of an adaptive control architecture comprises the decision procedures that are invoked to process specific requests or events.

Decision procedures may be invoked within a bandwidth broker a number of points. Following authentication of an incoming request, first authorization and then execution occur. Decision procedures may be invoked at both stages: for example, to determine whether a request should be granted, in the first instance, and to reallocate resources in the second instance if the newly authorized reservation over-subscribes available resources.

5.6.2 Bulk Transfer Support

So far, the proposed support of deadline file staging did assume either an adaptive application or the use of flow-based traffic shaping with the intention to pace TCP traffic. By configuring the average rate passing the traffic shaping mechanism of the rate below to the write frequency of the application, the router starts in queuing packets in the shaping queues until TCP becomes window limited. From there, the transmission is driven by the arrival of acknowledgments for data which has passed the traffic shaping mechanism. We can therefore control the transmission rate by traffic shaping.

Pacing TCP traffic in such an environment facilitates the simple use of network reservations even without the knowledge of the actual rate the application is writing data to the socket buffer. Using an oversized socket buffer together with the control to shape the traffic at the edge leverages TCP's self clocking feature to control the speed of transmission. In coordinating a reservation with the shaping rate, packet drops can be avoided and a well-defined throughput can be established. Hence, initiated by one of the reservation actuators a bulk transfer module is informed about the state change. This module is now able to use the policy propagation interface to update the configuration of the edge router for the bulk transfer flow.

5.7 Implementation Framework

An important issue for the provision of end-to-end guarantees is the availability of QoS-supporting mechanism in all domains that the traffic traverses. Several research and education networks have already started in planing the deployment Premium services based on the differentiated services architecture, such as the Dante Premium IP service [20]. A guaranteed capacity service [124] is another focal point of evaluation for research and education network. Following the "easy-to-deploy paradigm" this dissertation evolves a QoS

architecture which is not relying on the existence of a broad variance of aggregate behaviors. The proposal still claims to use the lowest service level required to address the service demand, but it is capable of supporting a heterogeneous mixture of flows by two services which can be built on top of a single aggregate behavior:

- A Premium service for delay-sensitive applications, and
- a GR service for elastic application, i.e. TCP-based applications.

The fundamental concept is to combine the expedited Per-Hop Behavior with the ability to engineer the traffic as this is offered by MPLS. The need for traffic engineering (TE) was already listed in Section 4.2. In this context, TE-mechanisms are also used to address potential conflicts in a single aggregate.

The evolved implementation framework consists of the following building blocks:

- Each router uses a packet classifier for incoming traffic on each interface. The ingress router of the source domain performs packet classification based on the quintuple $(IP-source, Port-source, IP-dest, Port-dest, Protocol)$, while other routers perform packet classification based on the DSCP. Note that for an MPLS domain, the ingress router uses the DSCP to map aggregates to LSPs. Core routers of an MPLS domain apply EXP-inferred LSPs.

- A token bucket mechanism is used on the ingress ports of edge routers to police incoming flows for which bandwidth is reserved. This ensures that the arrival curve of each service requesting flow is constrained by a token bucket. For transient domains, the incoming traffic aggregate is policed against the current state of the Service Level Agreement. Nichols et. al. [97] state that the token bucket (see below for a formal definition) used for implementing a Premium service is only allowed to fill to the maximum packet size. To allow bursts without packet drops, it uses holding queues on the input interface of the ingress router, which queue Premium packets until a token can be used to transmit the data. However, the policing concept proposed here is not based on the assumption of holding queues. The reason for this is that some commodity hardware is not supporting such mechanisms. Instead, exceeding packets are either dropped explicitly, or—when drop precedences are configured in the core—marked differently. Note that the use of shaping queues on the output interface of the ingress router of the source domain is designed to provide the functionality of holding queues for elastic applications.

- A packet marker is used to mark packets which successfully passed the policer.

- A congestion management mechanism is used on each output interface of the domain. While congestion avoidance could, in principle be used to set up a GR service — by setting the WRED best-effort maximum threshold below to the minimum threshold of the related aggregate — an explicit scheduling mechanism allows a better trade off between bandwidth assignments and queuing. In a multi-aggregate environment, WRED is an option to extend the propound queuing configuration. The proposed implementation framework distinguishes between two different types of queuing mechanisms:

  - Class Based Weighted Fair Queuing (WFQ) is used on the egress port of edge routers for TCP traffic or non delay-sensitive UDP/TCP flows, i.e. to schedule the packets of a GR service. It might also be used at core routers, but for the achievement of stronger delay boundaries an EF implementation should dedicate as much bandwidth to the EF queue as possible. WFQ ensures that in periods of congestion—namely, when packets get queued in the router because the output link does not provide the capacity for delivering them immediately—each IP service class receives at least the fraction of the output bandwidth defined by the weight for that class. When the interface is not congested, queues can in principle use any available bandwidth, regardless of their weight.

  - Non-preemptive Priority Queuing (PQ) — also called low-latency queuing — is recommended on both, the egress port of edge routers and on core router, with the intention to support delay-sensitive UDP flows. When a mixture of delay-sensitive traffic and non delay-sensitive traffic is handled by a single EF aggregate behavior, a service differentiation should be provided by the edge router. Whenever this is possible, the implementation should rely on additional packet marking capabilities of the router and should establish a per-flow configuration of the output interface for the first-hop router.

- Traffic shaping is used on the egress interface of the first-hop router to shape out short-term bursts to the contracted rate. Note that traffic shaping is proposed to be performed on a per-flow base and can thus perform the equivalent function of holding queues on the ingress site of the network which is a requirement for supporting bursty traffic in a single aggregate.

Figure 5.12 illustrates the proposed configuration for a single aggregate.

Additionally, this dissertation proposes to control the allocation of core tunnels by MPLS traffic engineering mechanisms in transient domains. Whenever an aggregated reservation is instantiated, external routing algorithms can be incorporated to setup a Label Switched Path (LSP) in which the core tunnel is placed. Lakshman et al. [78] have shown that the optimal routing calculation is NP hard. To ensure that the control procedures finish within a reasonable time scale, heuristics or approximation algorithms are appropriate.

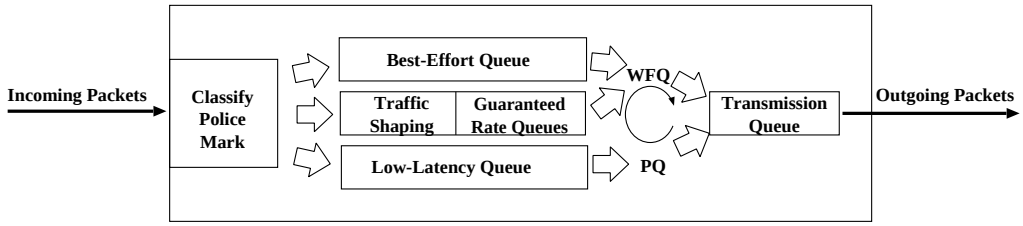

Figure 5.12: Proposed configuration of the ingress router's output interface. A router internal packet classification mechanism is used to differentiate between the Priority traffic and the individual GR requests.

The evolved formalism gives a criteria for the appropriateness of an LSP placement candidate. The proposed framework is therefore open for a range of these external routing algorithms, depending on the intention of the domain owner. In contrast to offline-routing algorithms [79], the propound bandwidth broker allows to incorporate an external routing algorithm that does not depend on the assumption that all LSPs are known in advance. However, the incorporated advance reservation capabilities improve the knowledge about future resource assignments which can be used for planning ahead.

The drawback of this dynamic environment is the potential impact of new LSP candidates with the service parameters of existing LSPs. Re-routing of existing LSPs is a solution for this problem, but it has to be done in a careful manner. MPLS itself offers features to support the update of an LSP such as path protection. A problem here is, however, the existence of potentially mis-ordered packets within a single aggregate. Mis-ordered packets might cause TCP to fall into its congestion control mode and thus to reduce the actual goodput. To avoid this problem, the allocation of LSPs should consider a potential service decrease when further service requests are granted by assigning less capabilities to the RAR than currently assured.

It is likely to happen that service parameters will also contain availability and link error parameters. The protection capabilities are one of the key-features of MPLS. It is thus clear that the on-line routing algorithms should also consider the use of these mechanisms to enhance their services under failure conditions by a graceful service degradation. Note that the proposed network monitoring must be able to recognize those events.

MPLS also supports the weighted balancing between multiple parallel tunnels. However, this feature also has the problem of potentially mis-ordered packets. The assignment of trunks should therefore always be on a single LSP using a well-defined path. However, different services can be mapped to different paths.

5.8 Conclusion

This chapter presented the design of an advanced bandwidth broker which addresses the particular needs of the Grid. The evolved architecture consists of five fundamental building blocks:

An external signaling interface that is responsible for the receipt of service requests and for the propagation of feedback information. It interfaces with external entities. To embed the bandwidth broker into the more general advance reservation framework of the Grid, the external signaling interface is decoupled from the Grid resource management system by a process, which intermediates between the two interfaces.

A State Repository that is responsible for the bookkeeping of all reservations and that is thus the fundamental building block for the ability to reserve network resources in advance. It incorporates the knowledge about the topology of the domain, including any routing information. To support the dynamic mapping of aggregated service requests, the State Repository uses the abstraction of a traffic trunk. A traffic trunk connects ingress and egress routers with a particular level of service, and is thus the base for incorporating traffic engineering mechanisms into the bandwidth broker by mapping the traffic trunk explicitly to the underlying topology.

A Policy Repository that is acting as the clearinghouse for admission control rules. In addition to intra-domain rules, it also contains the Service Level Specifications with peered domains. These Service Level Specifications are a representation of the diverse trust relationships and usage policies that can apply in a complex multi-domain Grid environment.

An internal signaling interface that is responsible for interfacing with the routers. It propagates the current QoS policy information, monitors the edge-devices for reservation events, and supervises the domain for events related to the State or Policy Repository.

Control Procedures that are driven by the signaling interfaces. Using the repositories they process reservation events and trigger the appropriate reaction such as initiating a configuration update through the internal signaling interface or responding to the requester through the external signaling interface.

To incorporate the bandwidth broker into the Grid resource management framework, this chapter presented two different approaches. In the first approach, multi-domain network reservations were treated as a special case of any coordinated reservation and allocation of multiple Grid resources. In this scenario, the requester is responsible for providing the required authorization information. While this concept is appropriate for tightly-coupled Grid environments with a direct trust relationship between all intermediate and possible end-domains, it is not feasible for a more complex environment, where service providers or national research networks are responsible for service provisioning. The architecture therefore allows for a second approach, in which network reservations were interpreted as

a single reservation step which is controlled by the bandwidth broker of the source domain. The proposed security model provides the secure transport of requests from source domain to destination domain, with each bandwidth broker on the path being able to enforce local policies and to modify the request with additional constraints.

The evolved architecture is designed to offer the demanded services, namely a Premium and a Guaranteed Rate service, by a careful combination of standard Differentiated Services building blocks, even in an environment with a single defined aggregate behavior: the Expedited Forwarding (EF) Per-Hop Behavior. In this context, the proposal extends the functionality of the ingress router of the source domain by an additional packet differentiation within a single aggregate. This mechanism is also used to improve the support of adaptive applications by integrating per-flow based traffic shaping as rate adaptation mechanism within the control procedures of the bandwidth broker.

The potential drawback of reduced service capabilities for complex aggregates is addressed by incorporating traffic engineering mechanisms into the design. Traffic engineering algorithms are proposed for aggregated reservations in transient domains, i.e. reservations which can be simultaneously used by multiple authorized subjects, without the need to signal the use of each individual reservation in transient domains. To improve the effectiveness of this, an RSVP-style request signaling was introduced in which an aggregated reservation is encoded by a domain specific Differentiated Services Coding Point. The feasibility of this approach is extended by the ability to build the demanded services in domains offering a single better-than best-effort aggregate. Finally, using the network calculus, this chapter derived a set of formulas which give a delay-boundary for a path candidate of a traffic engineering algorithm.

Chapter 6

Design Evaluation

This chapter evaluates the proposed architecture of the Grid-enabled bandwidth broker that provides network Quality of Service for Grid applications. It first presents a prototype implementation with the intention to demonstrate the technical feasibility of the proposed framework. Finally, it presents a thorough experimental evaluation of the claimed capabilities for different implementations of Differentiated Services architecture.

6.1 GARA: A Prototype for a Grid Bandwidth Broker

This section describes the bandwidth broker implementation of the General-purpose Architecture for Reservation and Allocation (GARA).

6.1.1 Overview

The General-purpose Architecture for Reservation and Allocation (GARA) provides advance reservations and end-to-end management for quality of service on different types of Grid resources. GARA is a research prototype of the Globus Project [51, 50], a project that is developing fundamental technologies needed to build computational grids, and is an advance reservation architecture for computational grids.

A GARA system is composed of a number of *resource managers* that each implement reservation, control, and monitoring operations for a specific resource. Resource managers have been implemented for a variety of resource types. A bandwidth broker is the resource manager of particular interest here. Uniform interfaces allow applications to express QoS needs for different types of resources in similar ways, thereby simplifying the development of end-to-end QoS management strategies. Mechanisms provided by the Globus

Toolkit [50] are used for secure authentication and authorization of all requests to resource managers. A directory service allows applications to discover resource properties such as current and future availability.

Figure 6.1 illustrates an architectural overview of GARA which is a three-tier architecture. Access to grid resources is provided by a specific Grid service which can be viewed as a mapping procedure between the grid resource management semantic and that one implemented by the underlying local resource manager. In this scenarios, a user creates a reservation using the GARA client software. GARA uses the standard resource allocation protocol in Globus to securely communicate with the reservation service (in this case, a network reservation service), which offers a uniform view of a resource that can be reserved. This Grid service translates the service request to a concrete resource allocation request of a particular resource manager. Once the request is propagated to the local resource manager, it is handled by the local resource manager, the local resource manager interacts with the resources it controls in order to ensure the reservation is enforced. An example here would be an update of the configuration of an edge router.

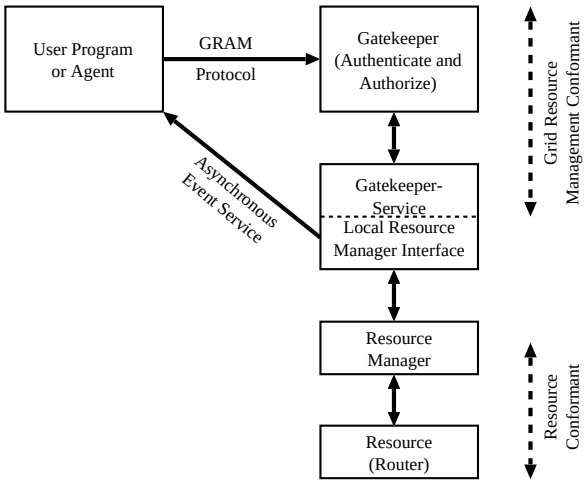

Figure 6.1: An overview of the multi-layered architecture of GARA. End-users use an interface which is compatible with the Globus Resource Allocation Manager (GRAM) [27] protocol to communicate with a particular Gatekeeper-service. This service translates the reservation requests to the particular resource manager interface and offers notification services.

Each service layer of GARA is represented by a separate Application Programming Interface (API) that allows users and applications to manipulate reservations on different layer of abstractions. On the Grid resource management level, for example, essentially the same calls are used to make an immediate or advance reservation of a network or computing resource. Once a reservation is made, an opaque object called a reservation handle is returned that allows the calling program to modify, cancel, and monitor the reservation. Other functions allow reservations to be monitored by polling or through a callback mechanism in which a user's function is called every time the state of the reservation changes in an interesting way. For the proposed bandwidth broker design, a callback is handled

by the reservation actuator which uses the notification interface to propagate events to the requester.

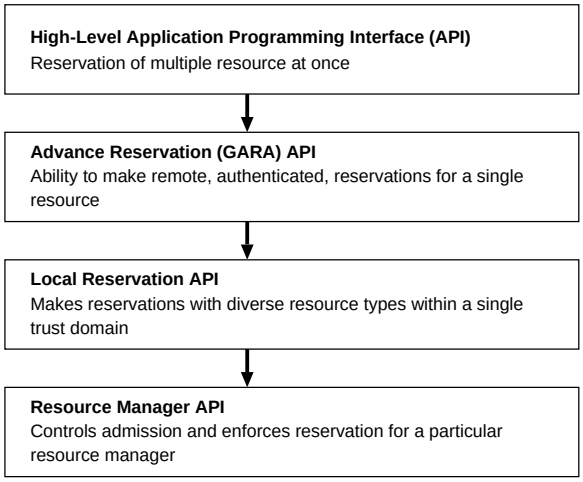

Figure 6.2: The multi-layered architecture of GARA consists of application programming interfaces. Each layer is offering a distinct level of abstraction.

Figure 6.2 illustrates the layered structure of these APIs. The Local Reservation and Allocation Manager (LRAM) API provides direct access to reservation functions within a trust domain, while the remote API provides remote access to LRAM functionality, addressing issues of authentication and authorization in a Grid environment. Both APIs implement the functionality described in the preceding paragraph.

The uniform treatment of reservations provided by GARA makes it possible to define and reuse co-reservation and co-allocation libraries that encode strategies for the coordinated use of multiple resources [28]. Because different resources (e.g., computers and storage systems) can be manipulated via the same function calls, standard libraries can be developed that encode common functionalities such as fault recovery strategies.

One co-reservation library that was developed implements an end-to-end network API that provides end-to-end analogs of each of the remote API calls. This API follows the source-domain-based signaling approach and allows the user to create, monitor, cancel, etc., network co-reservations: that is, reservations involving more than one network resource. This API allows users and applications to ignore details of the underlying network topology. Figure 6.3 illustrates the use of this end-to-end API. The functionality of this API relies on an existing Grid service of the Globus Toolkit, the metacomputing directory service [47], which is the repository for the contact addresses of Grid resources and services and is used by the API to identify all relevant bandwidth brokers.

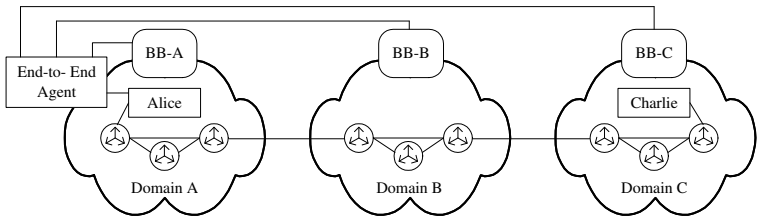

Figure 6.3: Illustration of the use of GARA's high-level end-to-end Application Programming Interface (API). Here, a network reservation is treated as a co-reservation. The API assures that all relevant bandwidth brokers are contacted for a network reservation. Note that this mechanism relies on a directory service in which Grid resources including bandwidth brokers list their Uniform Resource Locators (URLs).

6.1.2 General Implementation

Each resource manager integrated into GARA provides admission control and reservation enforcement for the controlled set of resources. One of GARA's main goals is to incorporate existing resource managers whenever possible, in order to provide reservation capabilities without having to create them from scratch. However, several resource managers were created for GARA, because appropriate resource manager did not already exist when GARA was begun. This section describes some of the implementation details on how resource managers are integrated into GARA.

GARA uses a State Repository consisting of slot tables as illustrated in Figure 5.3, where each slot represents a single capacity delegation as a "slot" of time. These slot tables can be used by any resource manager to keep track of reservations. Therefore, this architecture complies to that proposed in Figure 5.4. In addition to the provision of basic slot table operations, the library also allows the subscription to slot table related event. Hence, it offers a basic implementation of the Repository Monitoring function listed in Figure 5.11. Requests to a resource manager are made via an additional interface layer, the Local Reservation and Allocation Manager (LRAM) API. The intention of this API is to map see service requests to a potentially remote resource manager. It offers common functions that add, modify, or cancel service requests; timer-based callbacks generate call-outs to resource-specific routines to enable and cancel reservations. Note that either only certain elements of this API need to updated to instantiate a new resource interface, or the LRAM API can be fully replaced by the API calls of the related resource manager. The GARA prototype supports a set of resource managers including the Distributed Soft Real-Time scheduler [24] for computing reservations, and the Distributed Parallel Storage System [127], for a dedicated access to network storage.

Access to reservation capabilities is provided by a particular API (also called the GARA API) which integrates the resource manager into the Grid. It uses the Globus toolkit which resource management framework is based on a component called the gatekeeper. This simple service accepts incoming SSL-encoded HyperText Transfer Protocol (HTTP) requests

to execute local services, performs authentication via calls to the Generic Security Service API [53], performs authorization, and if these checks succeed, dispatches the incoming request by invoking—in the case of GARA—a simple program that uses the LRAM API. Whenever the requester registers a callback, the newly invoked program does not terminate. Instead, it is accompanying the reservation during its lifetime, if the callback is not canceled explicitly.

The GARA prototype uses two "Grid" services provided by the Globus toolkit: the Globus Lightweight Directory Access Protocol (LDAP)-based information service for publishing reservation status information and for accessing path information, and the public-key based Grid Security Infrastructure for authentication and authorization services.

6.1.3 Bandwidth Broker Implementation

The particular design issues of GARA with respect to the coordinated reservation of multiple resources including a basic bandwidth broker functionality were addressed in [112]. This subsection advances the implementation of GARA's bandwidth broker.

GARA's bandwidth broker is implemented as a separate process which communicates to external entities by two different implementations of the External Signaling Interface. One is based on the asynchronous Nexus [54] communication library; the other one uses a secure socket communication. When started, the process first initializes the repositories by reading a set of configuration and state files. As illustrated in Figure 6.4, the State Repository uses slot tables for each individual link of the controlled domain. While these slot-tables are used in the admission control procedure, they are not stored persistently in the State Repository. Instead, they are dynamically maintained in memory. Traffic trunks, however, are persistently stored in the State Repository using a domain wide slot table of unlimited capacity.

Whenever a new request arrives, it is first mapped to a temporary traffic trunk. This trunk is then dynamically mapped to the underlying topology. Here, domain specific traffic engineering algorithms can easily be incorporated by replacing the shortest-path first algorithm of the current implementation. Once, a path was selected, the admission control procedure proves whether all related slot tables provide the appropriate capacity. If this is the case, the temporary traffic trunk becomes persistent, i.e. it is synchronized to a state file. Whenever a traffic trunk becomes persistent, it is always associated with an Reservation Actuator. The Reservation Actuator is not implemented as a separate thread. Instead, it is incorporated into the main procedure of the bandwidth broker. It performs its monitoring task by registering an event notification at the Repository Monitoring thread for all relevant slot tables.

The Internal Signaling Interface of GARA's bandwidth broker is implemented as a separate process which automates a telnet [104] session to the related router. Once the telnet session

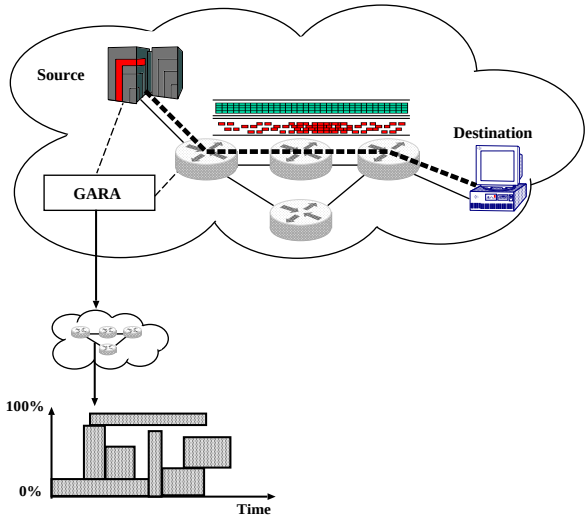

Figure 6.4: Illustration of GARA's implementation of a bandwidth broker. Requests are signaled through the standard GARA application programming interface and mapped to the External Signaling Interface. The State Repository has the knowledge about the underlying network topology and maintains a slot table per link and uses this state base for its admission control. The Reservation Actuator maintains an additional slot table per request, i.e. a traffic trunk.

was initiated, it propagates the state update by a sequence of configuration commands of the router's command-line interface. A secure out-of band connection to the edge routers is thus an important deployment aspect. A similar process is used to implement the proposed loss-rate sensor. This sensor, when activated for a particular flow, uses a separate thread to periodically execute a command which probes the router statistics. It then translates these statistics to a common notation and propagates the information back to the Reservation Actuator.

The Reservation Actuator is using a specific notification service to inform the user about the event. In this context, it uses a property of the intermediated Gatekeeper-service (Figure 6.1). Whenever a subject registers a callback for a reservation, the GARA implementation creates a socket for the user which is used to securely receive asynchronous events. In that case, the Gatekeeper-service does not terminate, but monitors the reservation and communicates to the user whenever the status of the reservation changes. Consequently, the event notification is implemented as a multi-layered system. On the Grid level, notification is done by the Gatekeeper-service, which follows the common Grid notification practice. On the bandwidth broker level, the External Signaling Interface uses an separate notification mechanism which is based on the asynchronous Nexus [54] communication library.

The routines of the Reservation Actuators are also linked with a special bulk transfer Decision Procedure. The intention of this Decision Procedure is to support deadline file-staging by applying for the minimum reservation required to meet the deadline and by additionally incorporating unused EF resources. The latter is accomplished by the interaction between

all Reservation Actuators and the bulk transfer Decision Procedure. When a bulk transfer reservation arrives, the Reservation Actuator extracts the current state of the domain from its State Repository and determines the amount of unused capacity over the particular trunk. Note that the proposed architecture does not require to address the issue of bulk transfers in transient domains. Here, bulk transfers are placed in core tunnels, i.e. they use unused capacity of core tunnels. Consequently, this service is provisioned by the bandwidth broker of the source domain.

Whenever the Reservation Actuator has reserved the available amount of unused bandwidth, it marks it as background reservation, i.e. as a preemptable reservation, and links it with a bulk transfer decision procedure. Any further change of active reservations signals this event to the background Reservation Actuator. It then uses the bulk transfer decision procedure to perform the following actions:

- Reducing the minimum reservation based on the time-interval the additional background reservation was active.
- Recalculating the new amount of available bandwidth based on the new state

The proposed loss-rate sensor described in Section 5.6 is implemented and used to determine the bandwidth reservation required to support a particular UDP flow. The motivation for this use of adaptation is that many application developers have no knowledge or QoS mechanisms or of the principles by which QoS parameters are determined.

When indicated by the requester, the Reservation Actuator is starting a separate thread which is using the Internal Signaling Interface to periodically pull the fraction of dropped packets for this particular flow. Let P be this amount and D the overall amount of packets of this flow; hence, $\frac{D-P}{D}$ is the fraction of packets that conformed to the reservation. Whenever P is exceeding a specified threshold, which is in many cases 0, the loss-rate sensor signals the two values P and D to the related reservation actuator. The reservation actuator uses this information to calculate what reservation it would have needed to make such that no packets would have been dropped, as follows:

$$R_n \frac{D-P}{D} = R_o$$

or

$$R_n = \frac{DR_o}{D-P},$$

where R_o is the old reservation and R_n is the new reservation.

While it is possible to determine the attempted transmission rate of a UDP flow based on the ration of dropped and exceeded packets, a comparable adaptive strategy for TCP is significantly more difficult. Data that an application attempts to write into a socket buffer may not be copied immediately because TCP's sliding window protocol does not allow to free previously written data. Also, TCP slows its sending rate when it believes it has encountered congestion. (In our case, TCP has not encountered congestion, but an aggressive QoS policing mechanism.) Nevertheless, TCP is used extensively in the applications that interest us, and so it is important to support TCP if we can.

Consider the behavior of TCP when a reservation is made that is smaller than the rate that it is sending. At the beginning the transmission rate is limited by the congestion window controlled by TCP's slow-start phase. Because it is actually less than the reserved rate, the token bucket will get filled. Once the congestion window corresponds to the rate the flow has reserved, subsequent transmissions will be able to exceed the reserved rate until the token bucket has emptied. At this point, the router starts to drop all packets exceeding the actual rate limit. Depending on the reserved rate and the achieved congestion window size, multiple packets will be dropped in a single round. This is because we can assume for most scenarios that the first rounds of the slow-start phase will provide a full token bucket, therefore the congestion window can increase by more than one.

TCP reacts to these packet drops by either the selective or fast retransmission followed by the congestion-control phase. Whenever to many packets were dropped, the protocol stack experiences a timeout. In the latter case, TCP is entering the slow-start phase again. This round, TCP switches over to the congestion-control phase, once half of the former congestion window size is reached. Hence, the window size is increased linearly until the token bucket is empty again.

We can expect that these cycles form a specific pattern, in which each single round reaches an application/reservation specific congestion window until packets will get dropped. Assuming that the application provides sufficient data to the socket buffer at any time—which is equivalent to the fact that the attempted transmission rate exceeds the maximum possible rate of the network at any time—we can illustrate this pattern.

Figure 6.5 visualizes the behavior of a Reno TCP flow with delayed acknowledgments when timeouts occur. In the first round-trip round, it will only submit one (some implementations start with 2) segment. With every (delayed) acknowledgment it will increase the congestion window size which results in an increase of packets transmitted during the next round-trip. Having reached half of the maximum window size, TCP will switch to congestion control phase. The increase now is much smaller. Once the flow exceeds the configured rate limit, it will still increase its congestion window size, until the token bucket is empty.

To formalize this behavior we introduce the following definitions: Let R be the window size (in segments) corresponding to the reserved rate and the round-trip time of the net-

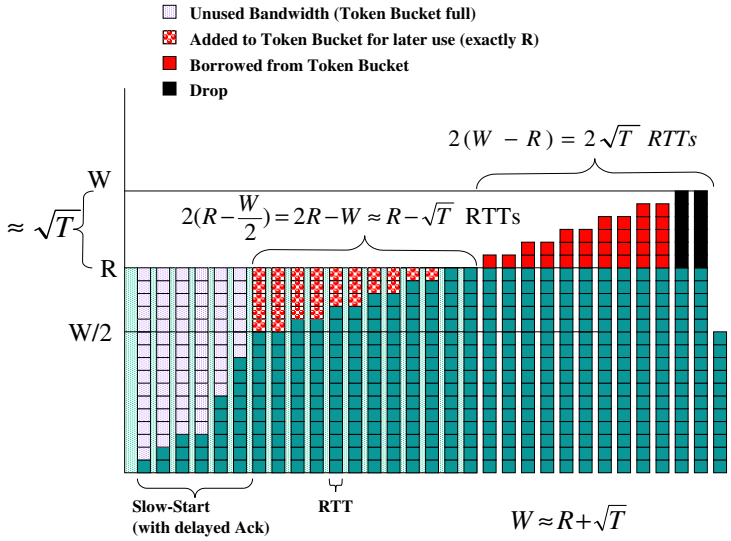

Figure 6.5: The behavior of a TCP flow when it is exceeding its reservation. The illustration swapped the tokens added to the token bucket with the unused bandwidth for readability reasons. In reality, the token bucket is first filled in a slow-start period.

work [141]:

$$R := \frac{ReservedRate}{MSS} * RTT$$

Furthermore, let T be the token bucket depth in segments and W be the maximum window-size reached (note: a larger window-size will be reached at the beginning of the flow). The value of W can be calculated from the equation:

$$T = (\sum_{i=1}^{W-1-R} 2i)$$

Solving this we get:

$$W = R + \frac{1}{2} + \sqrt{T + \frac{1}{4}} \sim R + \sqrt{T}$$

This equation allows to express the assumptions made about the token bucket depth:

$$R > \sqrt{T}$$

However, when trying to develop a heuristic for determining the attempted rate based on

these cycles, several limitations occur:

- When TCP times out, the actual timeout value is hard to predict. It depends on the configuration of the system and the TCP internal state.
- The ability to recover the lost packets by some retransmission mechanism strongly depends on the TCP implementation of the end-systems.
- The router statistics are typically updated very rarely (in the order of 10 seconds).
- The quality of the heuristic depends on both: the sampling interval and the round-trip time.

Because of these difficulties the application based adaption is not an appropriate solution for TCP flows, even when it is implemented as a binary search for the required reservation [112, 56]. A flow-based use traffic shaping is the correct vehicle to assure that the reservation conforming transmission of TCP flows in a Guaranteed Rate environment. The loss-rate sensor is therefore exclusively used for Premium UDP-based flows.

6.2 Experimental Setup

This section discusses at set of experiments designed to examine different Differentiated Services implementation strategies which are addressing the specific requirements of Grid applications.

6.2.1 Testbeds

The experimental configuration, illustrated in Figure 6.6 and figure 6.7, comprises a testbed at Argonne National Laboratory (the Globus Advance Reservation Network Testbed: GARNET) connected to a number of remote sites, including Lawrence Berkeley National Laboratory (LBNL), and a testbed at the Forschungszentrum Jülich GmbH. Connectivity between GARNET and LBNL is provided by the Energy Sciences Network (ESnet) DS testbed.

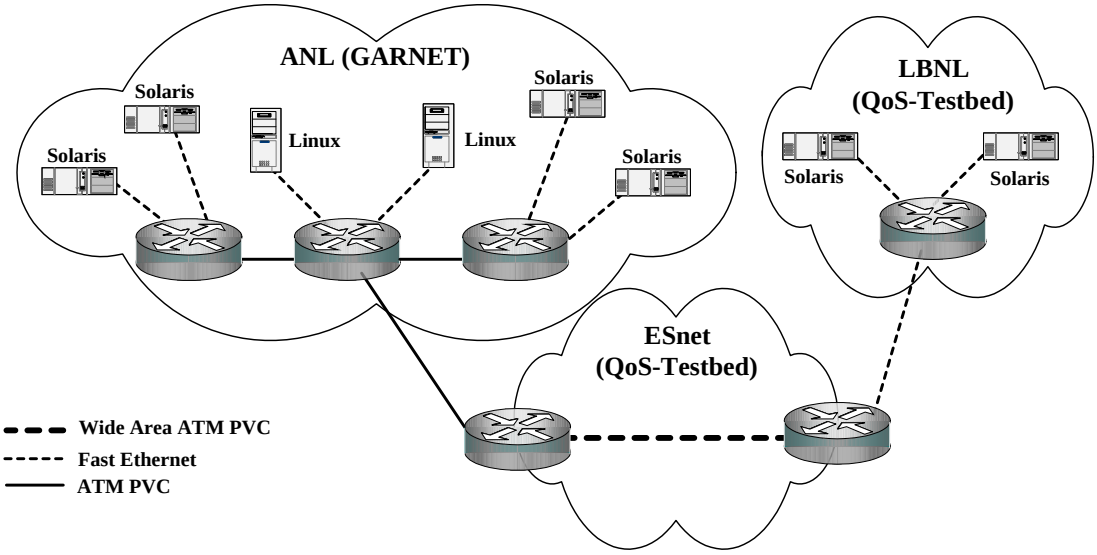

Figure 6.6: The GARNET testbed at Argonne National Laboratory and its connection to Lawrence Berkeley National Laboratory over the Energy Sciences network (ESnet). End-systems are connected via Fast Ethernet, routers via ATM PVCs.

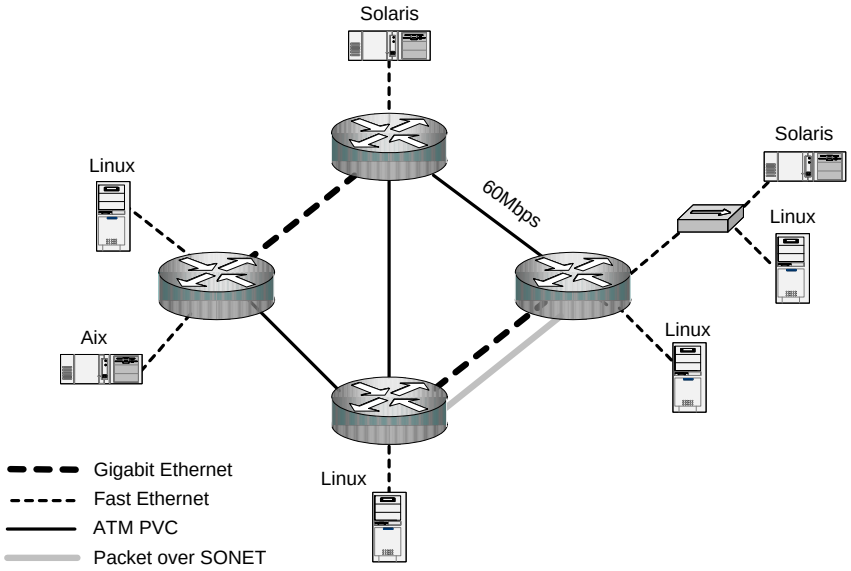

Figure 6.7: The testbed at Forschungszentrum Jülich GmbH. The Variable Bit Rate Non Real-Time PVC link is the bottleneck.

Cisco Systems routers are used for all experiments. GARNET's 7507 series routers are connected by Optical Carrier 3 (OC3) ATM connections; across wide-area links, they are connected by PVCs of varying capacity. End-system computers are connected to routers by either switched Fast Ethernet or OC3 connections. The Cisco Systems 7204 series routers

at Jülich are either connected by OC3 ATM connections, by Fast Ethernet, by Packet Over SONET (POS), or by Gigabit Ethernet connections. End-system computers are connected to routers by switched Fast Ethernet connections. Hence, the minimum maximum transfer unit (MTU) size of both testbeds is that of the end-systems: 1500 byte.

6.2.2 Evaluation Tools

The analysis required appropriate measurement methods. First, it was necessary to use a tool that was capable of producing a network flow (TCP and/or UDP) with specified characteristics such as bandwidth or burstiness. Second, a tool for creating a well-defined amount of competing traffic was needed, to create some congestion on the internal network, as well as a tool measuring the delay.

- *TCP stream generator:* A TCP traffic generator was evolved which is capable of generating a flow with a predetermined rate. This generator operates by adapting the frequency of write() calls to achieve the desired rate. Note that this adapts only the frequency with which the transmitter fills the socket buffer, and not the actual transmission rate. The host's TCP stack is still responsible for transmitting those packets onto the network media, by using TCP's flow and congestion control mechanisms. However, this behavior reflects exactly the challenge of QoS-aware TCP communication.

- *Delay-sensitive UDP traffic generator:* This traffic generator (created by Brian Adamson, Naval Research Laboratory, and Sean Gallavan, University of Notre Dame) consists of a set of programs that provide the ability to perform IP network performance measurements using UDP/IP unicast and multicast traffic. The toolkit generates real-time traffic patterns so that the network can be loaded in a variety of ways. Because each packet contains a time-stamp, this tool can easily be used to analyze Type-P-one-way delay [4] and jitter, if clocks are synchronized.

- *rude/crude – Delay-sensitive UDP traffic generator.* This traffic generator generates real-time traffic patterns so that the network can be loaded in a variety of ways. Packet sizes and transmission rates for individual information flows can be controlled and varied using script files. Because each packet contains a time-stamp, this tool can easily be used to analyze the one-way delay [4] and delay jitter. To create a feasible traffic pattern, the experiments used script files generated from publicly available video traces [46]. The injected traffic was fragmented by applying IP fragmentation for the transmission of frames that exceeded the MTU, which we consider as being allowed here, since we configured the DS classes to prevent from dropping fragments.

- *Best-effort UDP traffic generator:* This traffic generator (originally created by Andy Adamson, University of Michigan) divides the flow into intervals of one second, in which UDP datagrams are transmitted at fixed frequency until a given per-second rate is achieved. This dissertation improved its stability to provide a configurable constant bit rate by adding real-time library timers to the existing code. Having achieved the desired amount of data per second, the transmission stops until the next interval starts. Hence, the tool can be used to introduce burst periods of up to one second.

6.3 Multi-Aggregate Environment

This section discusses a set of experiments designed to validate possible differentiated services scenarios when multiple per-hop behaviors are applied. It concludes with a recommendation for a service class assignment.

6.3.1 Implementation and Evaluation of the Best-Effort Service

Applying the plain best-effort service to two example applications, an application which is representing a delay sensitive video-conferencing scenario and a background congestion traffic, we generate the baseline for our evaluation. The configuration applied allocates the remaining capacity of the ATM Interface, which is not used by any other class, to the BE class. In the following experiments no other class than BE is used, resulting in an assignment of 60 Mbps of the bottleneck ATM link to the BE class.

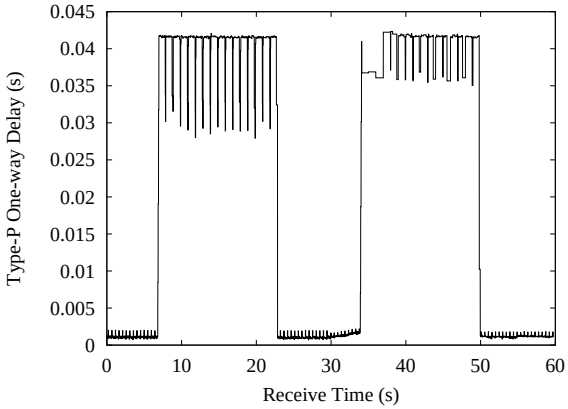

Figure 6.8: Data transmission profile of an MPEG-4 encoded TV-news sequence showing two periods of congestion. Each competing best-effort flow lasted about 15 seconds and. Note that no service differentiation was applied.

The tx-ring-limit parameter on the ATM interface card that specifies the queue size, which is assigned to the applied ATM virtual circuit, was set to 16 particles each of 512 byte allowing to store up to four MTU on the ATM interface. This value is by far smaller than the default value, but it has to be applied to allow for an efficient QoS implementation [44]. The BE layer 3 queue was configured to hold at most 256 packets. This queue size, which is a trade off between delay and loss rate, is considered as being feasible for BE traffic, which is rather sensitive to packet drops than to queuing delay in a range of a few tens of milliseconds.

In Figure 6.8 the delay measured when transmitting the news sequence in the BE class is shown. Congestion is generated by applying an UDP stream with two bursts, each of ten seconds duration.

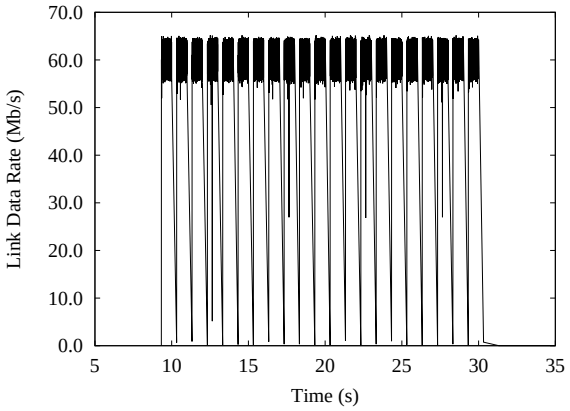

Figure 6.9: Transmission profile of the competing UDP best-effort flow which was causing congestion during its periods of bursts. All TCP flows of this section compete with this traffic profile.

As Figure 6.8 shows, the delay is bounded to about 42 milliseconds, showing some minor effects on the measurements due to tail-drop in the router. The delay corresponds to an effective data rate on the ATM interface of about 48 Mbps after subtracting the ATM induced overhead. While this delay is acceptable for streaming video applications, it can be critical for real-time video applications like video conferencing or remote haptic and tracking.

Next, the performance of TCP flows in the BE class is analyzed. The protocol specific properties of TCP were already discussed in Section 2.3.1 and 4.3.1 when downstream congestion was applied. Here, the presented analysis is refined by also taking a TCP upstream congestion into account. Note that a network reservation is uni-directional. For TCP, however, there is always a bi-directional communication. It is therefore important to analyze the impact of upstream network congestion for a uni-directional file transfer.

Upstream congestion only affects the acknowledgments of the TCP stream, but the experiments demonstrate a major impact of the delay and the loss rate of the acknowledgments on TCP performance and the related traffic profile, which also has been addressed analytically by Hasegawa et al. [65] and in simulations by Wu and Williamson [140]. Congestion

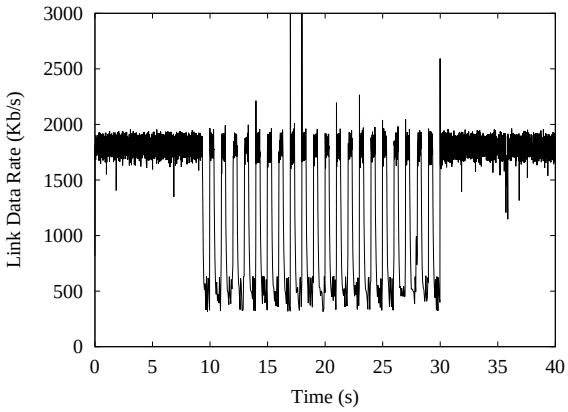

Figure 6.10: Throughput of a TCP file transfer when upstream congestion was applied. The impact caused by the bursty competing UDP traffic illustrated in Figure 6.9 is significant.

is generated by applying the UDP flow shown in Figure 6.9.

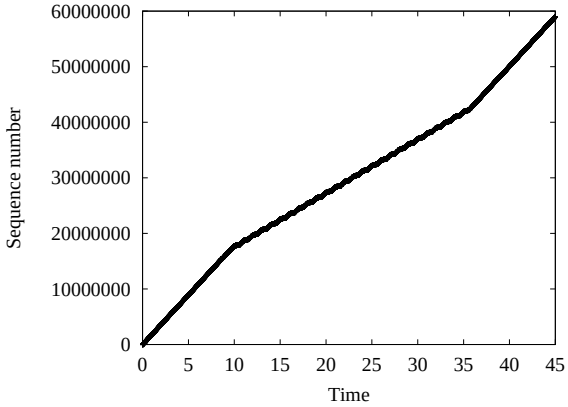

Figure 6.11: TCP sequence numbers over time of a TCP file transfer under upstream congestion. The slope of the curve indicates the achieved throughput. The impact caused by the bursty competing UDP traffic illustrated in Figure 6.9 can be clearly seen.

Figures 6.10 and 6.11 show the throughput and the sequence number over time. About 10 seconds after the start of the TCP transmission upstream congestion is generated by the UDP flow that is shown in Figure 6.9, leading to a significantly reduced throughput. The effects on the throughput are generated by an increase in the Round-Trip-Time (RTT) shown in Figure 6.12.

Besides acknowledgments can get delayed due to congestion they can also get lost. While the throughput impact of a lost acknowledgment is less significant, it makes the TCP stream more bursty.

Figure 6.13 shows the effect of an increasing RTT on the sequence number over the time. At first, the TCP transmission is bounded by the application, which writes at a speed of

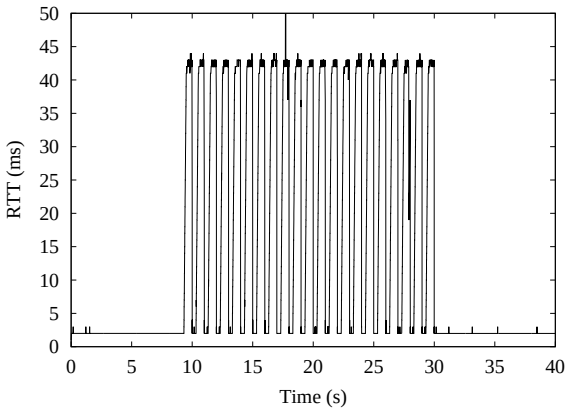

Figure 6.12: Actual round-trip time of packets of a TCP file transfer when upstream congestion was applied. Acknowledgments become queued because of the bursty competing UDP traffic which is illustrated in Figure 6.9.

14.5 Mb/s to the socket buffer, leading to a quite smooth transmission without actually using the maximum available window size. When the upstream congestion starts, the application is no longer the limiting factor. The available window slows the transmission down, as can be seen from the utilization of sequence numbers at the upper border of the available window in Figure 6.13.

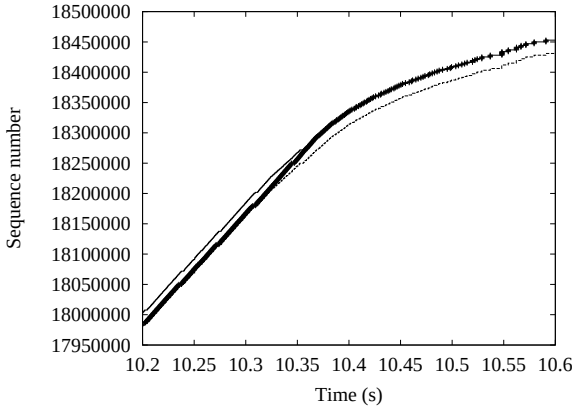

Figure 6.13: TCP sequence numbers over the time within the boundaries of TCP's sliding window. The slope of the curve indicates the achieved throughput. The scenario illustrates the behavior of the TCP flow at the beginning of an upstream congestion period.

Furthermore the decreasing gradient denotes a significantly reduced data rate. This reduction of the data rate R can be addressed analytically by Equation 6.1 with RTT denoting the round trip time and W denoting the window size.

$$R = W/RTT \tag{6.1}$$

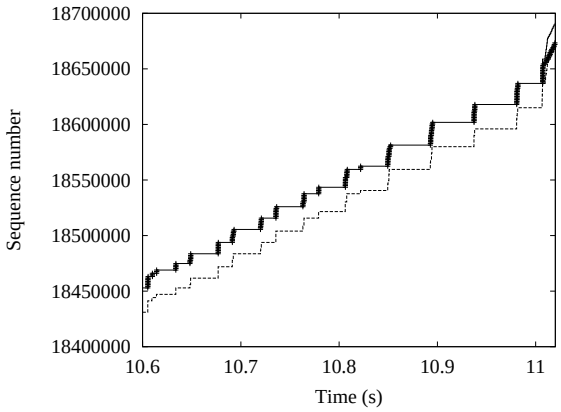

Figure 6.14: TCP sequence numbers over the time within the boundaries of TCP's sliding window. The slope of the curve indicates the achieved throughput. The scenario illustrates the significant burstiness of the TCP flow at the end of an upstream congestion period.

Clearly, Equation 6.1 shows that the problem of an increased RTT because of delayed acknowledgments can be addressed by an increase of the window size, but since the window scale option is only negotiated during connection establishment, any dynamic right-sizing is bounded to the specific window scale value of its connection. Unfortunately, most operating systems do not allow the explicit specification of the window scale option. Instead its value is derived from the socket buffer size during connection establishment. Whenever servers have to maintain a huge number of parallel TCP connections, a trade-off between the desired window size and the dedicated amount of kernel memory has to be made.

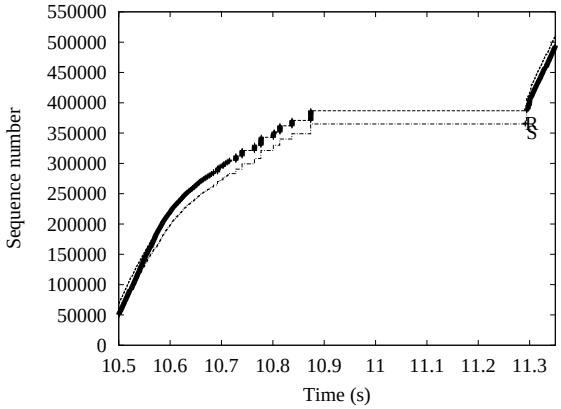

Figure 6.15: TCP sequence numbers over the time within the boundaries of TCP's sliding window. The slope of the curve indicates the achieved throughput. The scenario shows that the applied upstream congestion causes a TCP timeout.

Additionally, we can observe from Figure 6.14 that lost or delayed acknowledgments make the TCP flow significantly bursty. Especially upstream delay jitter increases this effect, as can be seen at the far right of Figure 6.14, when the congestion is resolved. In this case acknowledgments for a complete window can be received by the sender as an acknowledg-

ment burst, thus leading to huge downstream data bursts. This clustering of data segments might result in downstream congestion.

Of course, the effect is increased if bigger TCP windows are used and can even lead to downstream packet drops. Hence, the assumption of an upper bound for the window size of a single TCP connection is reasonable.

Another negative effect of acknowledgment bursts is shown in Figure 6.15. Here a complete burst of acknowledgments is locked out from queuing space and discarded, due to upstream congestion. Unfortunately TCP in this case has to wait for the retransmission timer to expire before it is allowed to perform a retransmission of one TCP segment. Furthermore TCP has to perform slow start after the timeout, reducing the transmission rate unnecessarily.

6.3.2 Implementation and Evaluation of an Olympic Service

A WFQ environment is used for the implementation of the Olympic service based on three AF PHB classes. Within these classes GARA is capable of managing the allocated resources and the relative load in order to allow for a service differentiation in terms of delay. The Olympic service [66] proposed by the IETF is realized by admission control and a class based over-provisioning. This subsection discusses a set of experiments with the transmission of the news sequence in each of the Olympic classes, with the classes configured according to Table 6.1.

Table 6.1 gives the configured percentage rate of the outgoing bottleneck ATM link, the capacity allocated for the different classes distinguishing between gross and net – with and without the estimated ATM overhead – and an over-provisioning factor indicating the relationship between the ingress and the core configuration.

Class	Percent	Gross Capacity	Approximate Net Capacity	Over-provisioning Factor
Bronze	5 %	3 Mb/s	2.4 Mb/s	≥ 1
Silver	10 %	6 Mb/s	4.8 Mb/s	≥ 2
Gold	15 %	9 Mb/s	7.2 Mb/s	≥ 3

Table 6.1: Core configuration of the Olympic classes. Excess traffic is explicitly dropped for Gold and Silver.

Within each of the Olympic classes a differentiation of the drop probability for differently marked excess traffic can be performed by applying WRED. Nevertheless, it is considered that such an implementation in an over-provisioned class is harmful for the BE class. The conforming traffic is therefore marked green, while excess traffic is dropped in the over-provisioned classes, whereas excess traffic is marked as red in the Bronze class.

The layer 3 queue size of each of the three Olympic classes was configured to 128 packets in the WFQ environment. Consequently, the ingress meter and marker is based on a token bucket with a confirmed information rate of 2.4 Mbit/s for all Olympic classes thereby

leading to the over-provisioning factors given in Table 6.1. A confirmed burst size of 32 MTU is used at the ingress. This value is intentionally smaller than the queue size applied, to avoid packet drops in the Olympic classes within the network and also to avoid a high utilization of the queuing space and thus to reduce queuing delays. Besides it has to be noted that the WFQ queue sizes are configured in packets, which can be smaller than the MTU, whereas the confirmed burst size is configured in bytes. An excess burst size as proposed for the SRTCM [67] was not used for the following experiments.

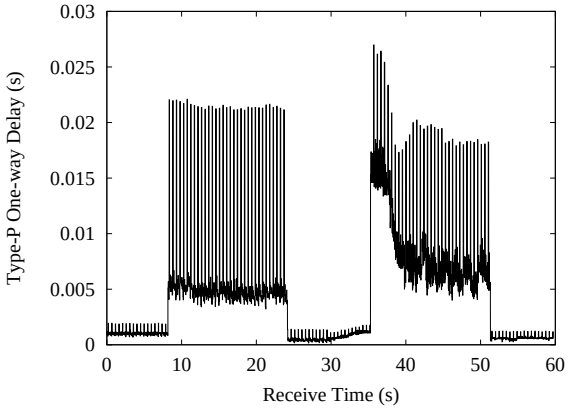

Figure 6.16: Traffic profile of an MPEG-4 encoded TV-news sequence when a DS Bronze service is used. Note that during two time intervals congestion was applied. Each competing best-effort flow lasted about 15 seconds.

Figure 6.16 shows the measured delay for the news sequence in the Bronze class and the impacts of congestion in the BE class on the Bronze class. Compared to the transmission of the sequence within the BE class, which is shown in Figure 6.8, the delay is reduced significantly. Furthermore packet drops did not occur in the Bronze class. Thereby AF based services can be applied as GR service without packet loss for conforming traffic.

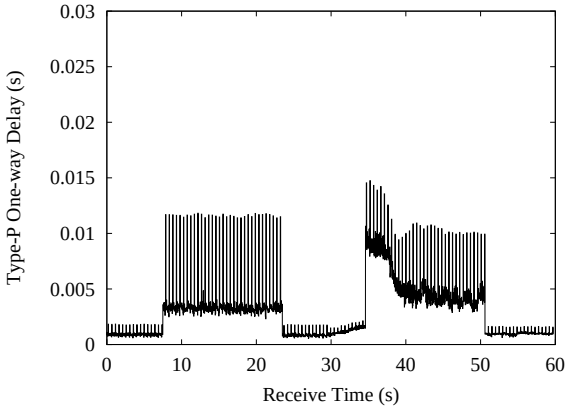

Figure 6.17: Traffic profile of an MPEG-4 encoded TV-news sequence when a DS Silver service is used. Note that during two time intervals congestion was applied. Each competing best-effort flow lasted about 15 seconds.

The delay and delay jitter differentiation, which can be achieved in addition by the Olympic service, is shown in Figure 6.17 and 6.18 for the Silver and the Gold class respectively, compared to the Bronze class in Figure 6.16.

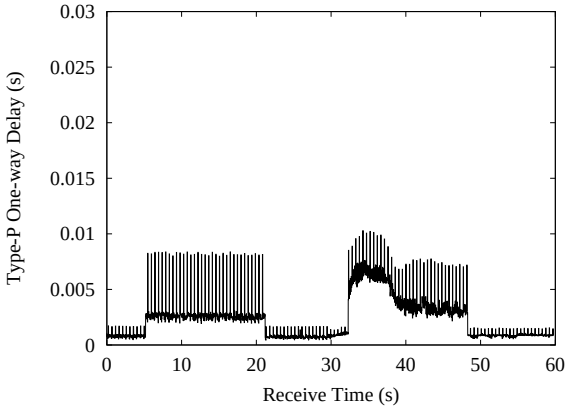

Figure 6.18: Traffic profile of an MPEG-4 encoded TV-news sequence when a DS Gold service is used. Note that during two time intervals congestion was applied. Each competing best-effort flow lasted about 15 seconds.

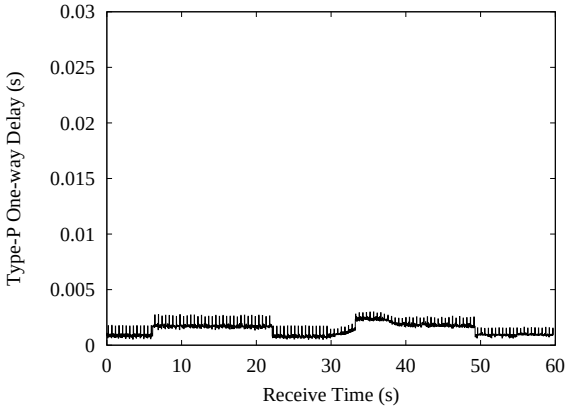

Figure 6.19: Traffic profile of an MPEG-4 encoded TV-news sequence when a DS Premium service is used. Note that during two time intervals congestion was applied. Each competing best-effort flow lasted about 15 seconds.

Additionally, we present experiments with TCP in the Bronze class and demonstrate how TCP can be configured in a guaranteed rate environment to achieve the desired throughput. We show that, if the pertaining class is configured properly, packet drops do not occur, which prevents from halving the congestion window, according to the Additive Increase Multiplicative Decrease technique implemented by TCP. The data rate instead corresponds to the capacity allocated for the flow. To avoid effects on the RTT by an upstream congestion, the acknowledgments are also transmitted in the Bronze class. The effects of different classes for TCP data and acknowledgment packets have been addressed by Köhler and Schäfer [81] in simulations. Nevertheless, the impact of placing the acknowledgments in

an even better service than the Bronze service is rather limited, since the Bronze class already can provide low loss and bounded delay.

The maximum window size is in our experiments controlled by setting the socket buffer size. The resulting RTT can be computed according to (6.1) with W denoting the maximum window size and R denoting the configured capacity in this case. The RTT adjusts to the available or configured capacity and to the configured maximum window size.

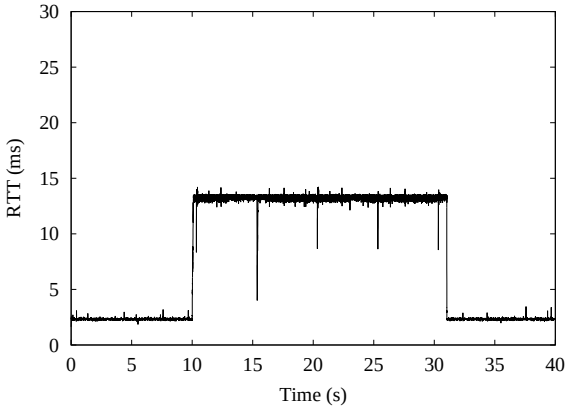

Figure 6.20: Actual round-trip time of packets of a Bronze TCP file transfer when downstream congestion was applied. The socket buffer size was 15 MTUs.

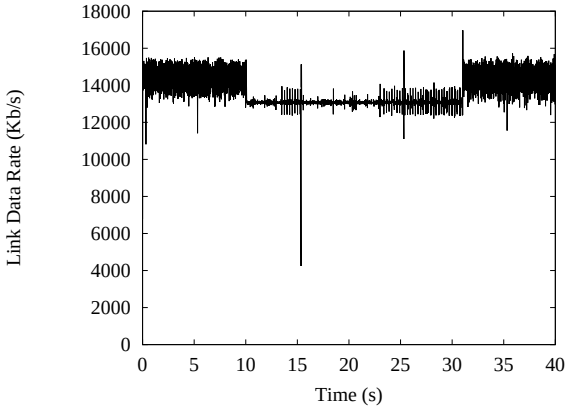

Figure 6.21: Throughput of a Bronze TCP file transfer when downstream congestion was applied. The link data rate was calculated based on the average throughput of 20 segments. The socket buffer size was 15 MTUs.

For these experiments the Bronze class was configured to 25 % of the bottleneck link capacity, corresponding to a net data rate of about 12.8 Mb/s. Figure 6.20 shows the RTT for a configured socket buffer of 15 MTU. Congestion in the BE class starts after 10 s and leads to an increase in the RTT, which corresponds to the queuing delay added by queuing the data of a complete TCP window.

Figure 6.21 shows the corresponding throughput. At the beginning the application limits

the data rate to about 14.5 Mb/s and after the BE downstream congestion started, the limitation is given by the configured capacity for the Bronze class at about 12.8 Mb/s and from the TCP point of view leads to a limitation of the sending rate by the offered window.

The same effect on the throughput can be observed, if the maximum window is increased by configuring a socket buffer of 32 MTU. Figure 6.22 and Figure 6.23 show the resulting RTT and throughput for this configuration. Again the RTT adjusts to the available capacity and the window size, being about twice as high as in the experiment with a 15 MTU socket buffer.

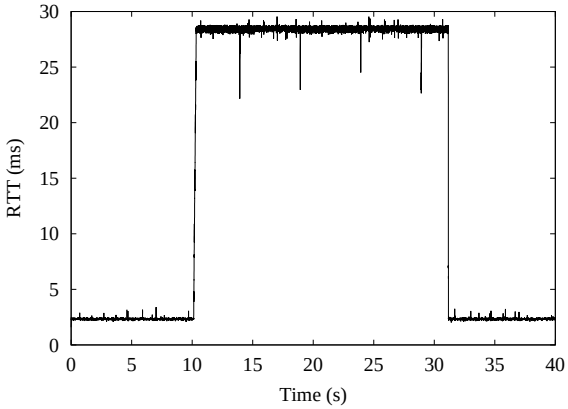

Figure 6.22: Actual round-trip time of packets of a Bronze TCP file transfer when downstream congestion was applied. The socket buffer size was 32 MTUs.

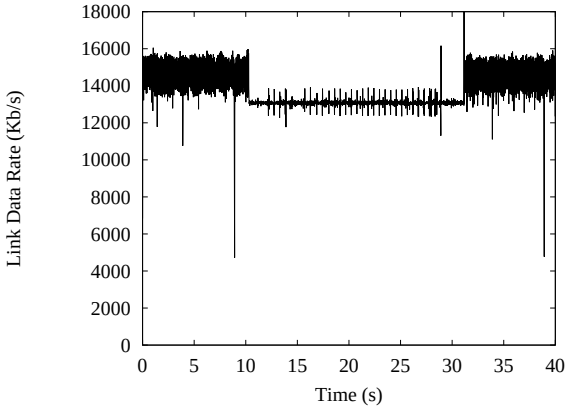

Figure 6.23: Throughput of a Bronze TCP file transfer when downstream congestion was applied. The link data rate was calculated based on the average throughput of 20 segments. The socket buffer size was 32 MTUs.

From the TCP experiment described above, it can be seen that WFQ acts as a traffic shaper in the absence of other flows in the same class. In general such a traffic shaping functionality should be located at the ingress side of a DS domain and should be per-flow based, whereas the core configuration is strictly class based. The traffic shaping can be performed

by a leaky bucket. On the other hand side a traffic shaping would prevent the TCP flow from utilizing unused capacity during times when the BE class is not loaded. Yeom addresses this problem by a two-window TCP and a mapping on drop precedence levels implemented by WRED [141], but unfortunately this approach requires a modification of the TCP protocol and its implementations.

6.3.3 Implementation and Evaluation of a Premium Service

The EF PHB can according to the relevant IETF standard [77] be implemented by a variety of queue scheduling mechanisms, among these are non-preemptive Priority Queuing (PQ), and class-based Weighted Fair Queuing (WFQ). Since commodity products of CISCO are used for our implementation, both mechanisms are a possible choice. Because of the fact that WFQ cannot easily be used for the proposed premium service of [97] on the 7200 platform, the Premium Service was implemented based on strict PQ.

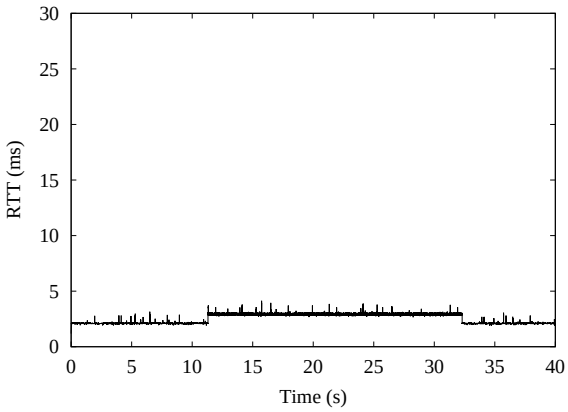

Figure 6.24: Actual round-trip time of packets of an EF TCP file transfer when downstream congestion was applied. The socket buffer size was 15 MTUs.

The ingress router was configured to apply a meter and marker with a confirmed information rate of 4.8 Mbps and a burst size of 32 MTU. Excess traffic is dropped. The parameters that were applied at the ingress router were reflected by the core configuration. The PQ scheduler was bound to 10% of the bottleneck link capacity, corresponding to about 4.8 Mbps after subtracting the estimated ATM overhead and bursts of up to 48 KB are permitted by the core configuration.

Figure 6.19 shows the results of a transmission of the news sequence for the premium service. A reduction of the transmission delay and delay jitter especially for big frames, which lead to packet bursts, becomes obvious for PQ compared to WFQ. In the configuration of the service the tx-ring-limit parameter, which is used to configure the outgoing interface queuing capacity, is of major importance [44].

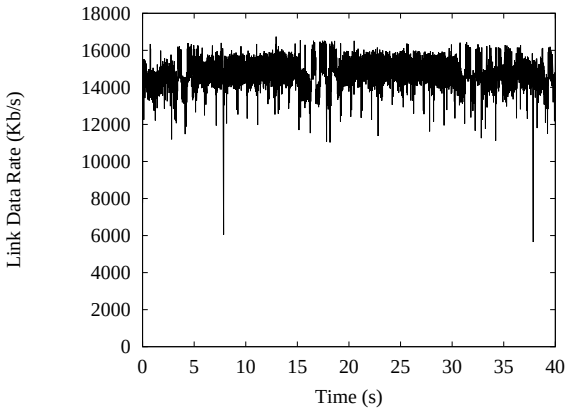

Figure 6.25: Throughput of a TCP file transfer using the EF aggregate when downstream congestion was applied. The link data rate was calculated based on the average throughput of 20 segments. The socket buffer size was 15 MTUs.

Figure 6.25 illustrates the achieved throughput for the EF aggregate. As before, the network was heavily congested between 10s and 30s. While the WFQ-based AF scenarios illustrated in Figure 6.21 and 6.23 indicated a traffic shaping, we do not see any impact on the achieved throughput in this scenario. Figure 6.24 illustrates the round-trip time. The minor impact is caused by the heavily loaded data link layer queue.

6.3.4 Proposed Assignment of Service Classes

Table 2.1 lists the various flows of a future teleimmersion application. We interpret this set of flows as an example unicast scenario which we want to support by our implementation of DS.

Service Class	Net Capacity	Max Bottleneck Delay	Type of Flow
Best-Effort	28.8 Mb/s	111.7 ms	Majority of the Internet Traffic [97]
Bronze	2.4 Mb/s	161.2 ms	Text Channel, Database Updates
Silver	4.8 Mb/s	81.1 ms	Currently Unused
Gold	7.2 Mb/s	54.5 ms	Control Channel, Rendering, MPI, Video
Premium	4.8 Mb/s	5.1 ms	Tracking, Haptic, Audio

Table 6.2: Mapping of applications to services. The delay boundary is calculated for a single bottleneck link.

Table 6.2 shows a mapping of these flows to the proposed service classes. The maximum bottleneck delay d given for the different classes except BE is derived from Equation 6.2, whereas $b_{n,i}$ denotes the maximum burst size for each flow that traverses the bottleneck link, c_b is the allocated bottleneck capacity of the respective class, c_i denotes the capacity of the incoming links, d_a is the maximum delay added by layer 2 queuing, and d_p denotes

the delay added by preferred priority traffic, if any.

$$d = \sum_i \left(\frac{\sum_n b_{n,i}}{c_b} - \frac{\sum_n b_{n,i}}{c_i} \right) + d_a + d_p \tag{6.2}$$

The presented set of experiments applied $\sum_i \sum_n b_{n,i} = 32$ MTU, $\sum_i c_i \approx 98$ Mb/s, and $d_a = 1$ ms. The allocated capacity in Table 6.2 corresponds for each of the Olympic classes to the Peak Information Rate (PIR) [68], whereas the Committed Information Rate (CIR) is by the over-provisioning factor given in Table 6.1 smaller. In contradiction the given capacity of the premium service corresponds to the CIR. The premium service can by the means of PQ utilize the complete bottleneck link capacity for a configured burst size. The EF standard [77] requests that the aggregates maximum arrival rate is less than the minimum departure rate at any transit node to prevent from queuing. In order to implement such a controlled PQ environment, ingress premium traffic shaping is highly advisable [97], but it has to be noted that traffic shaping for a target PIR adds additional delay, especially in case of bursty traffic. Applying it reduces the maximum bottleneck queuing delay to $d_a = 1$ ms, but for a PIR of 48 Mb/s it leads to a maximum ingress shaping delay of 4.1 ms, whereas for a PIR of 24 Mb/s it leads to 12.1 ms, assuming an incoming fast Ethernets link with 98 Mb/s and applying Equation 6.2. Besides it has to be noted that scenarios can be constructed in which an arbitrary queuing delay at core nodes can arise even if the conditions of [77, 97] are fulfilled [9].

In general, it is recommended to apply the service, which matches the application requirements best, without exceeding them unnecessarily. Hence, video traffic requiring an upper delay boundary of 100 ms is placed in the Gold class, in order to reserve the premium class for more demanding flows. In the absence of traffic shaping a premium video flow creates unwanted premium bursts in the network, whereas video traffic shaping adds a noticeable delay.

6.4 Single-Aggregate Environment

This section extends the evaluation by analyzing the potential impact of TCP flows sharing a single Expedited Forwarding aggregate with delay-sensitive real-time flows.

6.4.1 Evaluation of Traffic Policing

This subsection gives an overview about a use scenario of the policing function offered by commodity hardware with respect to TCP flows.

Short-Term TCP Transfers

As stated above, TCP reacts when dropped packets are detected. If the transmitter recognizes a lost packet, it falls into either the congestion control phase or the slow start phase, depending on the number of contiguous packets lost. This behavior dramatically affects TCP performance. The first experiment was designed to illustrate this impact, by demonstrating the behavior of a relatively small TCP transfer (16.7 MB), with different transmission rates in combination with a token bucket based policing.

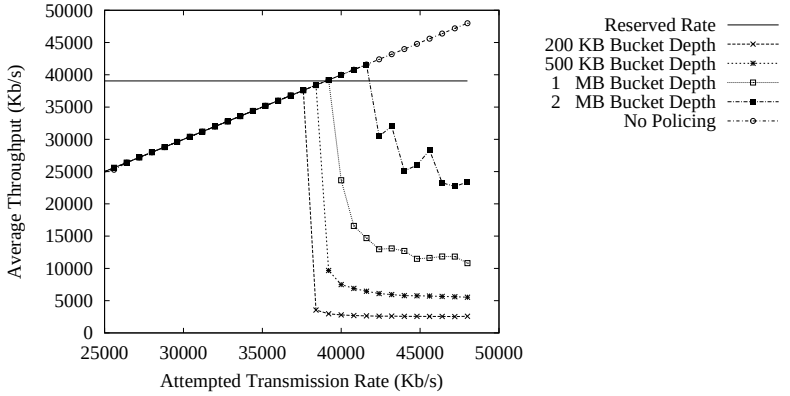

Figure 6.26: Average throughput of different short-term TCP streams with token bucket based policing on the ingress site. The token bucket depth was varied between 200 KB and 2 MB. The rate limit used for all tests was constant.

The results displayed in Figure 6.26 were gathered over several sessions by transmitting a TCP stream at a prespecified rate and altering the token bucket configuration. The impact of the token bucket depth is recognizable. Increasing the depth results in a higher short-term bandwidth, until the bucket gets empty and packets get dropped. Note that the increase of the bandwidth from 37500 Kb/s to 41500 Kb/s corresponds to the transmission time and the difference in the bucket depth. Having additional tokens for 1.8 MB allows a stream to exceed its limit by 40000 Kb/s for nearly 4 seconds. Overall, the transmitter is transferring 16.7 MB, which also can be done in less than 4 seconds. For an exact calculation it should be mentioned that the transmitter was using the path MTU discovery. For that reason the segment size used was 1460 bytes.

This experiment demonstrates effectively the behavior of TCP's slow-start feature. As soon as the token bucket is empty, the router starts dropping packets and often drops consecutive packets. The TCP transmitter recognizes that consecutive packets are lost and reacts to this by shrinking the congestion window to two, which has an enormous short-term impact on the actual throughput.

From this, one can conclude that exceeding the actual rate limit causes TCP to decrease performance drastically. Transmitting packets faster than the QoS rate limit has a negative impact on overall performance. Note that TCP might submit a whole socket buffer in one

burst as permitted by the offered receiver window and the congestion window. The actual token bucket configuration must be able to handle those bursts without dropping packets.

Long-Term Transfers

Besides the short-term behavior of TCP streams, it is important to analyze a stream with a longer duration. For that reason several long-term TCP sessions were monitored. Figure 6.27 represents an example for the behavior of a TCP session exceeding its rate limit. In this case the normally constant throughput starts oscillating. As soon as the policer starts dropping packets, the transmitter reacts with TCP's slow-start feature. This reduces the transmission rate drastically. As the router's token bucket begins filled again, the transmitter is able to exceed its limits again (if enforced by the application) and hence resumes its oscillating behavior.

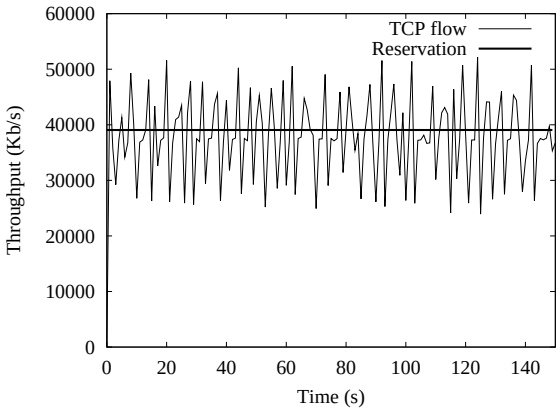

Figure 6.27: Throughput of a TCP stream with an underestimated token bucket policing on the ingress site. The attempted transmission rate was 50 Mb/s. The configured token bucket depth was 2 MB.

The conclusion is that exceeding the actual rate limit causes long-term TCP traffic to oscillate. This is mainly because of the slow start/congestion avoidance feature and the bucket depth. The greater the bucket depth, the greater the variation.

6.4.2 Evaluation of Heterogeneous Aggregates

This section presents an implementation of the Expedited Forwarding (EF) Per-Hop Behavior with the goal to analyze the behavior of a heterogeneous aggregate. It uses Weighted-Fair Queuing (WFQ) to emulate strict priority queuing by provisioning 99% of the bandwidth to the EF aggregate. This is to ensure that the queue of the EF aggregate caused by possible bursts allowed by the token bucket depth was minimized, i.e. the delay is minimized. Note that the bandwidth broker prototype GARA performs a careful admission control and is thus preventing the starvation of the best-effort traffic.

The particular challenge is caused by applying two services on top of a single EF aggregate: a Premium service for delay-sensitive applications and a Guaranteed Rate (GR) service for TCP applications. We now demonstrate the problem and present that our architectural design is appropriate for addressing the related issues.

Implementing the Per-Hop Behavior

The following series of experiments consists of three flows injected into GARNET 6.6 by Fast Ethernet connected end-systems, all passing a bottleneck ATM link of 100 Mb/s capacity (including all overheads):

- The first flow entering the testbed was a delay-sensitive Premium UDP flow. It ran longer than the other flows. GARA acted as a bandwidth broker to associate the flow to the premium class of the evaluated implementation. The related UDP traffic generator was configured to achieve a rate of 40 Mpbs by constantly submitting 1 KB packets every 0.2 milliseconds. The receiver continuously reported the delay calculated from the time-stamps in the packets.
- The second flow in the experiment was a GR TCP flow. Emulating an MPI-based distributed supercomputing application, we created a bursty TCP stream which was injecting data in chunks of 256 KB. The average rate of the flow was 16 Mb/s. Using GARA, we claimed a slightly higher guaranteed bandwidth reservation, allowing bursts of up to one full chunks. The guaranteed bandwidth TCP stream was the first one terminating.
- The third flow started during the experiment was a best-effort UDP flow. Our main intention was to create a heavy congestion by submitting 750 byte packets at a frequency of 10000 Hz, to achieve a rate of 60 Mb/s. To demonstrate the impact of a single premium flow under congestion, the competing UDP flow was still active after the TCP flow ends. The best-effort flow thus consumed a significant amount of the available bandwidth.

Figure 6.28 shows that the selected single-aggregate implementation is not appropriate for providing delay-sensitive services when bursty TCP flows use the same aggregate in parallel. If a burst introduced by the TCP flow exceeds the available output link capacity, packets get queued on the IP-layer queue. Because packets of the premium UDP flow are also queued, the delay variation increases significantly. As WFQ was configured to provide the full bandwidth, i.e. 99%, to the Premium class, the best-effort UDP flow starting as third flow does not have any major impact on the result. The minor impact demonstrated in Figure 6.28 is caused by a heavily loaded interface queues of the ATM interface, as the best-effort flow is permanently injecting competing traffic to the network.

It is important to note that the network administrator has the opportunity to configure the size maximum size of this transmission queue,in the Cisco implementation tx-ring-limit,

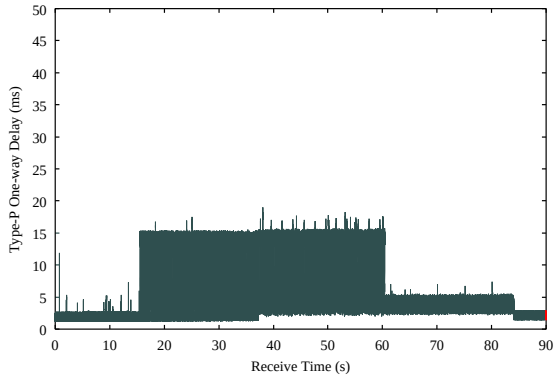

Figure 6.28: Type-P one-way delay for a premium UDP flow under congestion on GARNET. The experiment demonstrates that a bursty TCP stream sharing the EF aggregate has a significant impact on the achievable service.

and can thus directly influence the delay under congestion (see Section 5.3 for formal details). In this series of experiments, we configured the interface queue to provide room for up to 18 particles, each 512 bytes.

We can easily re-validate the result illustrated in Figure 6.28 by some simple calculations. The Premium service is used by a UDP application which is transmitting data at a rate of 40 Mb/s. This application shares the aggregate with a TCP flow which is injecting bursts of 256 KB at the link speed of its Fast Ethernet interface, i.e. at a rate of 100 Mb/s. Hence, the 256 KB of data is entering the EF aggregate within 20 milliseconds. The total amount of data entering the EF aggregate in this time interval is thus 356 KB. Assuming an ATM overhead of 20%, the EF aggregate is served at a rate of 80 Mb/s. We thus know that 200 KB of EF data is leaving the router. Consequently, at the end of the interval there exists an EF queue of 156 KB. Note that new UDP premium packets still arrive constantly. These packets lead to an effective rate of 40 Mb/s at which the queue size is reduced. By applying the maximum queue size of 156 KB we receive a delay boundary of 15 milliseconds.

Aggregate-based Traffic Shaping

In order to inject a traffic profile which is conforming to the Service Level Agreement with the peered downstream domain, the egress router of a DS domain might be enforced to shape out the traffic of a whole aggregate. When this is applied to the scenario illustrated above, the impact caused by the bursts of the TCP streams might be amplified by the queuing introduced by traffic shaping. Hence it is important to clarify the impact of this mechanism when shaping is done on the whole class.

Figure 6.29 illustrates the impact of traffic shaping when it is performed for an aggregate. Traffic shaping can be viewed as an additional constraint which limits the EF capacity of the output link by shaping the rate to the given traffic profile. Hence, EF packets get queued

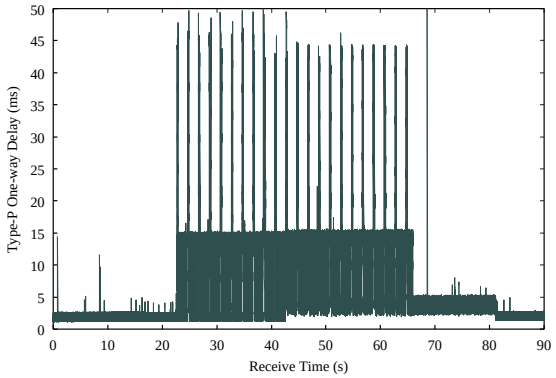

Figure 6.29: Type-P one-way delay for a premium UDP flow under congestion on GARNET. The experiment demonstrates that the impact of a bursty TCP stream sharing the EF aggregate is strengthened by aggregate based traffic shaping.

whenever a TCP bursts causes the shaper to become active. The experiment demonstrates that shaping is not implemented by a leaky bucket, i.e. a buffering mechanism for packets which "leaks" packets at a given rate. Instead, the underlying traffic shaping does allow some bursts. Traffic shaping is configured in terms of intervals, mean rates and bursts. During every interval, a maximum of burst size can be sent at any speed. However, the bit rate of the interface will not exceed the mean rate over any integral multiple of the interval. The experiment used the default interval length in, allowing most TCP bursts to pass without causing the shaping function to be activated. However, we still see several delay-spikes introduced by the queuing when traffic shaping became active.

This scenario leads to an EF PHB implementation that is even worse suited for the given mixture of flows than without traffic shaping. Note again that the scenario is realistic as domains might apply traffic shaping on their egress router. It also revalidates the thesis proposed in [115] that an EF PHB implementation supporting TCP flows without traffic shaping should provide as much of the Premium bandwidth under congestion as possible. The admission control has to ensure that the best-effort traffic is not starved.

Flow-based Traffic Shaping

As traffic shaping over an aggregate might have a negative impact on the delay variation of a Premium flow, the EF implementation proposed by the implementation framework presented in Section 5.7 proposed a flow-based service differentiation on the output interface of the egress router. In detail, the prototype bandwidth broker GARA applied an additional router internal packet marking mechanism, called "qos-group" which facilitates the an efficient packet classification. The router configuration propagated by GARA extended the basic DSCP marking by also assigning the "qos-group" for the related flow. This additional classification was then used to update the configuration of the output interface of the

ingress router to shape out the related GR flow.

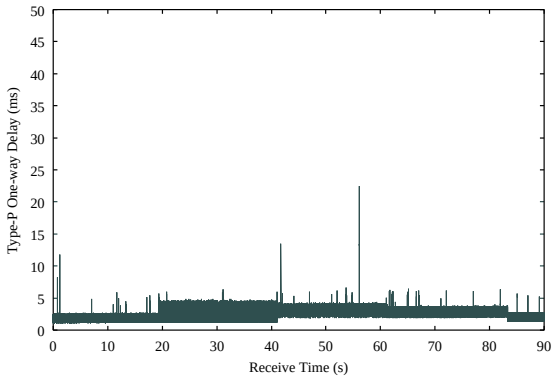

Figure 6.30: Type-P one-way delay for a premium UDP flow under congestion on GARNET. The experiment demonstrates that flow-based traffic shaping significantly reduces the impact of a bursty TCP stream sharing the EF aggregate.

Note that only the ingress router differentiates between those flow specific scheduling schemes. All core routers handle the premium aggregate based on the DS codepoint in a common way, assuming that the edge routers have eliminated possible bursts.

Figure 6.30 demonstrates the EF PHB implementation providing a premium service to TCP streams and delay-sensitive UDP flows. The remaining impact is caused by the interface queues, which are under the control of the network administrator. The figure demonstrates GARA's capability to support a mixture of flows in a single aggregate.

Figure 5.12 summarizes the way GARA configures the output interface of the ingress router. Beside the best-effort class, it uses non-preemptive priority queuing to serve UDP flows. Additionally, using the router internal packet marking, it applied traffic shaping with WFQ for the GR flow.

6.5 Applying MPLS

This section applies the Multiprotocol Label Switching architecture and summarizes the results of two important sets of experiments.

6.5.1 The Overhead of the On-demand Allocation of Tunnels

Whenever a transient domain decides to incorporate traffic engineering algorithms for the handling of core tunnel reservations, the question arises about the associated signaling overhead. To give a measure for this overhead in the specific environment of the testbed

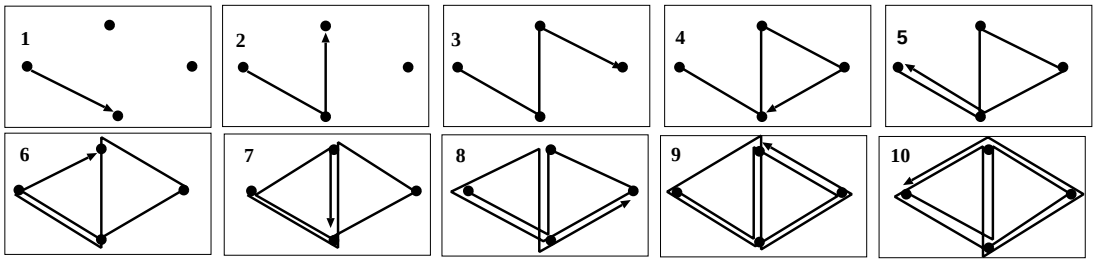

Figure 6.31: Logical view of the attempted placement of LSPs at the testbed in Jülich.

in Jülich (refer to Figure 6.7), we list an experiment which was performed by an accompanying diploma thesis [33]. The goal was to measure the time it takes to set up different explicitly routed Label Switched Paths. The mapping of the LSPs to the topology is illustrated in Figure 6.31. Because of the limited amount of routers in the testbed, some LSPs were extended to traverse a single router more than once by specifying different interface addresses in the explicit route object.

To measure the associated signaling overhead the experiment used the router internal logging functionality. Log entries were created with a timestamp whenever an LSP related event occurred. The resolution of the timestamp was one millisecond. Figure 6.32 lists the minimum, the maximum, and the average result of a set of 10 trials.

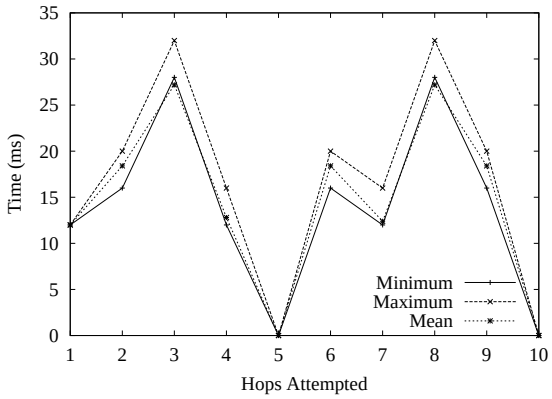

Figure 6.32: Overhead associated with the setup of LSPs at the testbed in Jülich. The decrease in overhead is caused by an optimization of the signaling procedures. Whenever the explicit route object lists interfaces of a single router more than once, it eliminates the related loop in between. Trial 5 and 10 did therefore not require any signaling at all.

The results of the experiment can be interpreted as follows:

- The Cisco implementation used the knowledge of the topology to eliminate loops. Loops in the explicit route object were recognized and eliminated.
- The time variation was caused by the ability to log the event within the operating system of the router and the resolution of the timestamp.

- The measured cost for setting up an LSP on the 7204 Cisco platform was varying between 5 and 8 milliseconds per traversed router. This cost is extended by a potential message propagation delay caused by the distance between peered routers.

It is very unlikely that an LSP will traverse over more than 20 different routers in a single domain. Hence, the signaling overhead associated with the allocation of an LSP appears to be passable. Especially in the context of advance reservation, the time constraint for the allocation of an LSP is relaxed. With immediate reservation, the time constraint can be relaxed either, because the LSP signaling can be done in parallel to the downstream propagation of the service request, without waiting for the positive response of the peered downstream bandwidth broker. Also note that the propagation delay of a telnet-based implementation of the Internal Signaling Interface is of the order of a second.

6.5.2 Link Protection

A major benefit of the use of MPLS is given by its failure protection capabilities. The fast-reroute feature allows to preserve the connectivity of LSPs for failures of a particular link, a particular node, or a more complex error condition. Especially link protection is designed to perform a quick restoration of connectivity. It also has the scalable property that the configuration of a single backup tunnel is able to protect all LSPs which are traversing the particular link.

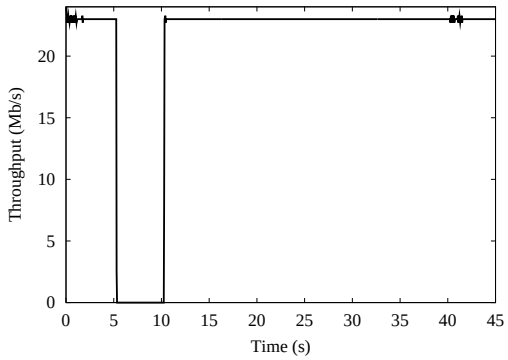

Figure 6.33: Service interruption caused by the time it takes to reestablish IP-convergence for a link failure situation. The curve illustrates the achieved rate over time. We recognize a service interruption at 5 seconds. Once IP-convergence was reestablished, the flow took was using the alternate Gigabit Ethernet link instead of the parallel Packet Over SONET (POS) link. Note that though the POS link was up again after 10 more seconds IP-routing remained on the alternate link for a while.

The experiments presented in this subsection explore the actual benefit of link protection compared to a failure situation occurred in an typical IP-domain which uses the popular Open Shortest Path First (OSPF) protocol as routing protocol. The layout of the two experiments is as follows:

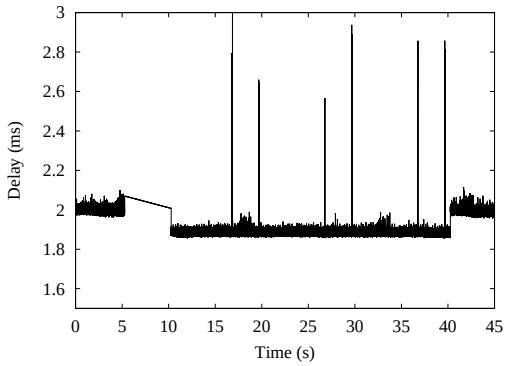

Figure 6.34: Service interruption caused by the time it takes to reestablish IP-convergence for a link failure situation. The curve shows the packet delay variation. In addition to the delay variation at the beginning, the curve indicates the phases when IP-routing was updated.

A constant bit rate stream was sending packets of the Ethernet MTU size (1500 bytes) every 500 microseconds. Around 5 seconds after the rude-application has started, the optical cable of the Packet Over SONET (POS) interface was manually unplugged. The crude-receiver was permanently reporting the receipt of packets to a log-file. A script processed the log-file and reported the achieved rate accumulated over 10 milliseconds, the experienced Type-P packet delay, and the accumulated amount of lost packets.

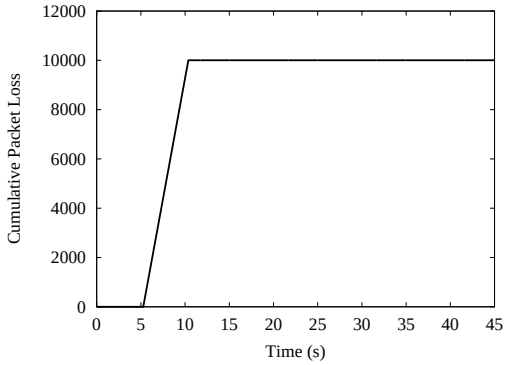

Figure 6.35: Service interruption caused by the time it takes to reestablish IP-convergence for a link failure situation. The curve shows the accumulated number of lost packets. At 5 seconds, the link failure occurred. IP-convergence took around 5 seconds, after which connectivity was reestablished. As the constant bit rate stream was sending packets with at a frequency of 2000Hz, the overall amount of lost packets was around 10000. No packets were lost when IP switched back to use the old link.

Figures 6.33 and 6.35 illustrate the achieved service when a standard OSPF configuration was deployed. The reason for the experienced disruption of connectivity is not caused by the layer 2 failure detection. The POS interface allows rapid failure detection by the explicit configuration of feedback mechanisms, which was done in the presented experiment. When a link state protocol is used to maintain the routing tables of each router, timers come into place which delay the recalculation of the shortest path. The intention of these timers is to avoid an extreme flooding of routing updates, with the goal to prevent

thrashing, especially when the link is flapping. It is exactly this timer which causes the significant disruption of service. Whenever this timer is reduced, it is done at the risk of an extreme overhead of CPU cycles of many routers. A significant decrease of the measured disruption is therefore not a realistic assumption.

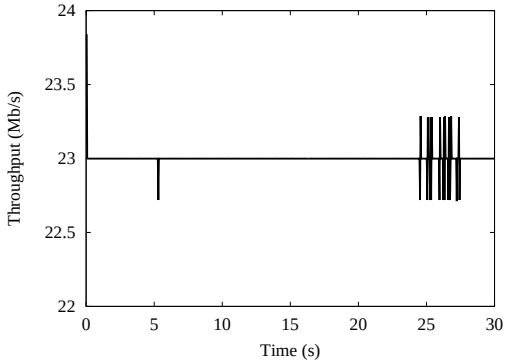

Figure 6.36: Service interruption in an MPLS domain when link protection is active. The curve shows the throughput over time. At 5 seconds, the link failure occurred. MPLS link protection was able to recover the failure situation quickly. Because the rate was averaged over 10 ms, the curve did not even drop significantly. Instead, the value indicates that only a single packet was lost. The graph shows a minor packet variance when the old link came back.

Figures 6.36 and 6.37 illustrate the achieved service when link protection was used. The edge router applied the efficient DSCP-based LSP assignment. Two different LSPs traversed the POS interface. A backup tunnel was configured to use the parallel Gigabit Ethernet interface whenever the POS connection is disrupted. The experienced disruption was negligible. Because the rate was averaged over 10 ms, the curve did not even drop drastically. Instead, the value indicates that only a single packet was lost.

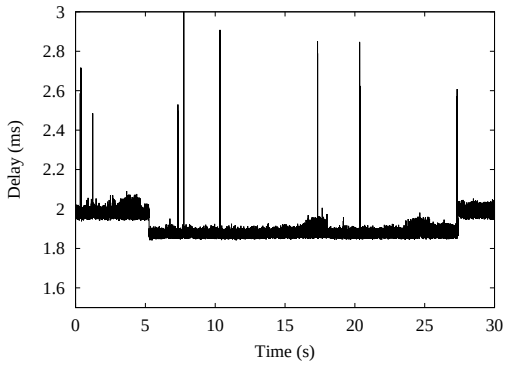

Figure 6.37: Service interruption in an MPLS domain when link protection is active. The curve illustrates the experienced delay over time. In addition to the delay variation at the beginning, the curve indicates the phases when link protection was updated.

Note that this experiment required the selected packet sized enforced MPLS to perform packet fragmentation on the backup tunnel. The MTU size of the related LSP was 1514 bytes. The POS interface exceeded this values, while the Gigabit Ethernet interface used

the MTU size of 1500 bytes. Because link protection is done by pushing an additional label to the label stack, the actual packet size was 1508 bytes. Hence, fragmentation was required for the MPLS case. Figure 6.37 indicates no additional delay compared to Figure 6.34.

The use of the protection capabilities of MPLS is an appropriate mechanism for a robust provisioning of network guarantees. When integrated into the decision process of LSPs candidates for external routing algorithms, a graceful service degradation is achievable.

6.6 Evaluation of Advanced Services

This section discusses a set of experiments that demonstrate the claimed capabilities of the proposed design.

6.6.1 Support of Bulk Transfers

An application-level adaptation procedure of high relevance implements deadline file staging by the use of the propose bulk transfer decision procedure. This procedure reacts on changes in reservation due to preemption by higher priority foreground flows (or termination of those flows) within the Guaranteed Rate (GR) service. It first signals the event to the subscribed application and secondly updates the bulk transfer related reservation. Whenever the application receives one of these notification events, it adapts its transmission rate with the goal of achieving throughput close to that bandwidth allocated to the bulk transfer flow.

The first experiment evaluates the ability to support multiple flows simultaneously and to support application monitoring of, and adaptation to, changes in reservation status. The experiment was conducted on the local GARNET testbed: see Figure 6.6. GARNET was configured to create a 45 Mb/s guaranteed rate channel in a 100 Mb/s network. It is based on five distinct flows: a bulk data transfer, operating as a "background" flow; a competing 80 Mb/s best-effort UDP flow (a traffic generator submitting 1,000 byte packets every 100 μsecs); and three independent, short-lived foreground flows with immediate reservations. In this and subsequent experiments we used a simple data transfer program, ttcp, as our "application." The GR flow and competing flows are sourced and sinked by different computers.

Figure 6.38 shows the bandwidth delivered to the foreground, background, and best-effort flows during the experiment. The procedure succeeded in delivering "excess" bandwidth to the bulk transfer application without comprising the foreground flows. The good bulk transfer performance achieved is made possible by the resource manager's callbacks to the bulk transfer application, which allow that application to change its sending rate in response

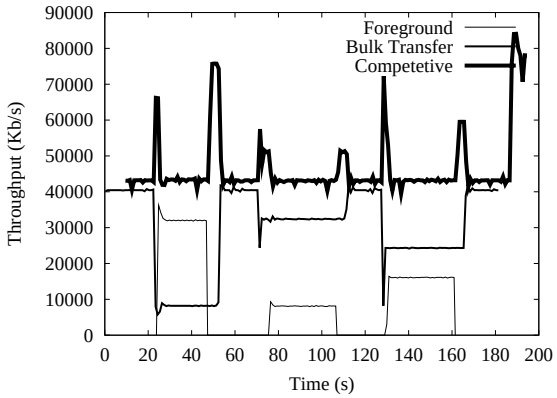

Figure 6.38: Throughput achieved for a mixture of Guaranteed Rate (GR) and best-effort services on GARNET. The rate is reported by the receiver on a per-second base. It is demonstrated that a bulk-transfer (background) application is able to exploit unused GR traffic without affecting foreground reservations.

to changes in its allocated bandwidth, hence avoiding packet drops and invocation of TCP slow-start. The following is a more detailed explanation of the graph:

1. The graph begins with the background TCP traffic, which has a bulk-transfer reservation. This flow is initially allocated 40.5 Mb/s GR bandwidth: that is, 90 percent of the 45 Mb/s GR traffic.

2. The competitive UDP traffic is started shortly after the bulk transfer but does not affect it due to the fact that the GR service was applied to the background traffic. flow.

3. At 25 secs, another application makes an immediate 36 Mb/s reservation and initiates a 32 Mb/s foreground flow. A callback notifies the bulk transfer application, which reduces its sending rate to adapt to the reduced reservation. (The and other similar transitions take a little time due to the time required to control the router.)

4. At 48 secs, the foreground application finishes its transmission and then cancels its reservation. Another callback allows the bulk transfer process to increase its sending rate to adapt to the newly available GR traffic.

5. Subsequently, two other foreground flows are created, with similar effects: a 9 Mb/s reservation (8 Mb/s flow) from 75 to 105 secs and an 18 Mb/s reservation (16 Mb/s flow) from 130 to 160 secs.

6. At time 185, the background flow completes and cancels its reservation. The competing traffic rate increases to its target of 80 Mb/s, actually exceeding this briefly because of the filled router queues.

Notice that each time the bulk transfer reservation is reduced, the bulk-transfer rate drops momentarily then recovers. This artifact is an indication for a fundamental problem of

application based adaptation of TCP flows in this scenario. By controlling the socket buffer write frequency, the injected traffic profile is dominated by the lack of data available for transmission, and not by the receipt of acknowledgments. We can therefore assume that the available window flow allow the immediate transmission of the written data. Hence, each single write typically results in a burst. The problem arises in each adaptation step. Because of the fact that the notification event is unidirectional, the application does not have the ability to synchronize the transmission rate adaption correctly with the updated policing configuration. Though the bandwidth broker is first informing the application about the new transmission rate before it instantiates the new reservation, there is no possibility to synchronize the operations more appropriately.

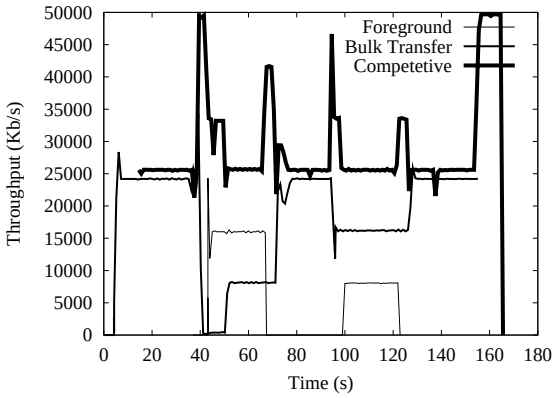

Figure 6.39: An example of bulk-transfer in using the ESnet wide-area testbed. The curves plot the throughput reported by the receiver on a per-second base.

Figure 6.39 shows results obtained in a wide area testbed between Argonne National Laboratory and Lawrence Berkeley National Laboratory. At about time 5, the (background) bulk transfer application began and was assigned all of the GR bandwidth. At approximately times 40 and 100, a foreground reservation began and the bulk transfer reservation was reduced. When the foreground reservations ended, the background reservation was increased. Notice that at time 15, competitive UDP traffic began but does not interfere with either the foreground or background reservations.

These results show that the application is successful in adapting the bulk transfer flow in response to information concerning preemption by foreground flows. Apart from a few artifacts, the bulk transfer flow maintains data transfer at a rate close to the amount of GR allocated to that flow. The artifacts are again an indication for the problem of application based rate adaption in this scenario. We therefore conclude, that we either need an application driven update of the background reservation, or, more preferably, a data rate pacing mechanism controlled by the bandwidth broker.

The model of bulk transfers used to date is intended to increase the economic use of the GR service. A highly relevant improvement of this scenario is to add a minimum reservation for the background traffic. By thus guaranteeing a particular amount of foreground traffic

to the background class, deadline staging operations can easily be implemented. Again, the economic use indicates that the required reservation to meet the deadline might change over time. Whenever a status update of the available bandwidth is received by the resource manager, it can easily calculate the new required amount of bandwidth.

The proposed architecture extended this model of bulk transfer by also applying for unused best-effort bandwidth. The required reservation of GR capacity to meet a deadline can in particular be decreased during the runtime of a bulk data transfer, if parallel sockets are used for the transmission. Thereby one of the sockets has to be configured to generate GR traffic at the confirmed rate to ensure that the deadline is met, whereas no reservation of network capacity is made for the remaining sockets, but these are mapped to a less than best-effort service in order to allow for fairness among competing and responsive best-effort flows. Within the QBone Internet2 project the DS Scavenger Service [123] is proposed to provide such a worse than best-effort service for high-bandwidth and also unresponsive flows. By applying the Scavenger Service the bulk transfer application not only collects unused GR bandwidth but also unused best-effort capacity at the price that the Scavenger Service can be starved by best-effort traffic. Whenever the bulk transfer application succeeds in transmitting an appreciable amount of additional data by applying the Scavenger Service, forward signaling of a reduced required GR rate to the reservation manager can be applied to decrease the reserved GR capacity and therefore reduce costs and allow for a smaller blocking rate of the reservation manager, due to a higher available capacity.

The prototype implementation of such a parallel bulk data transfer application is based on a simple data fragmentation and the use of two parallel sockets with the proposed mapping of one socket to the GR service and one to the Scavenger service. In order to overcome the effects of TCP congestion control in the Scavenger class, the implementation option of using multiple striped sockets for this class exists as this is offered by “gridftp” [2].

Figure 6.40 and 6.41 show results obtained from a similar testbed in Jülich which was identical except for the model of router. It demonstrates the capabilities of this combined use of a GR and a Scavenger service. A target deadline for a 280 megabyte (MB) file transfer of 200 seconds is applied. The required GR rate to ensure this deadline with a safety margin of 10 seconds is derived to 12 Mb/s, for which an initial reservation is made. The sender in addition to this GR service applies the Scavenger service in parallel, to utilize unused GR and best-effort capacity, if any is available. Each time the sender successfully transmits a configurable additional amount of data (25 MB) by applying the Scavenger service, it recomputes the reservation of GR capacity that is required to meet the deadline, and performs a reservation update.

In Figure 6.40 a scenario without additional traffic across an ATM bottleneck link of roughly 42 Mb/s net capacity is addressed. Besides the guaranteed rate of 12 Mb/s used by the GR stream, the Scavenger stream can use the remaining capacity with a rate of about 30 Mb/s. After 7 seconds, the sender performs the first GR service rate adaptation, due

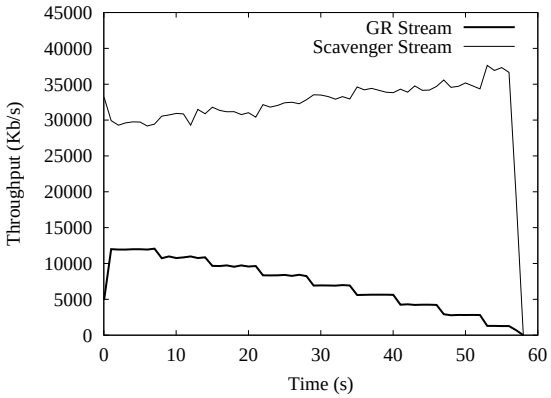

Figure 6.40: Throughput achieved with a combination of a GR service and a Scavenger service. The curves plot the rate reported by the receiver on a per-second base. The experiment demonstrates that a bulk-transfer application with a fixed deadline is able to use a GR service and in addition exploits unused GR and best-effort capacity.

to the additional 25 MB of data transfered by means of the Scavenger service, and lowers the GR capacity reservation to about 10.5 Mb/s. Since the ATM bottleneck link in this experiment is not used by any other flow, the rate at which the Scavenger service can be used increases proportionately. This process occurs repeatedly during the file transfer and allows a reservation manager to redistribute GR capacity. In addition the use of parallel sockets reduces the file transfer time to 58 seconds.

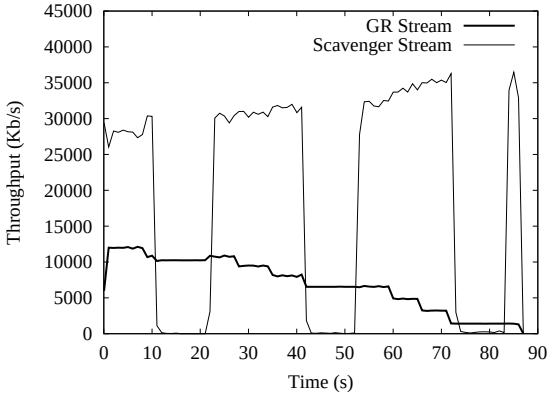

Figure 6.41: Throughput achieved with a combination of a GR service and a Scavenger service under congestion. The curves plot the rate reported by the receiver on a per-second base.

Figure 6.41 shows the same scenario under congestion. After 10, 40 and 70 seconds, congestion occurs, each time for 10 seconds. During these periods the congestion leads to the intended starvation of the worse than best-effort scavenger service and thus achieves the desired fairness among scavenger and best-effort flows. Nevertheless the deadline of the file transfer is never endangered, since the GR flow still receives the currently required guaranteed rate. Due to the reduced amount of data transmitted by applying the Scavenger

service, GR rate adaptations can in this scenario be made less frequently, and an overall file transfer time of 88 seconds is measured, which still is well below the GR flow target deadline of 190 seconds.

Thereby the use of parallel streams in a file transfer scenario and the mapping of these streams on a Guaranteed Rate and a Scavenger service achieves three main goals:

- The file transfer deadline is guaranteed.
- Excess GR and best-effort capacity can be used if available, to reduce the overall transmission time, and to be able to free GR resources earlier. Whenever access to higher-level services is associated with additional costs, the additional use of best-effort capacity reduces the cost overhead.
- Fairness towards the best-effort class is achieved by applying the worse than best-effort Scavenger service, thus allowing the coexistence of responsive best-effort traffic with high-bandwidth and also non-responsive Scavenger flows

We receive a very economic use scenario which is still fulfilling the desired capability.

A slightly different implementation of the parallel data transfer can alternatively be based on the DS Assured Forwarding, which is used to implement a GR service. Within one AF class a differentiation in terms of drop precedence can be applied to differently marked packets based on an implementation of WRED. At the ingress node of a DS domain a meter typically applies a token bucket mechanism such as the Two Rate Three Color Marker [68] proposed for the use with AF. This marker performs a marking of packets to be treated in the core with a low drop probability if the traffic confirms to a committed information rate and with a high drop probability, if it exceeds this rate.

6.6.2 Evaluation of TCP Pacing for a Guaranteed Rate Service

So far, we required the application to be instrumented to adapt to a given rate. In this section we evaluate the use of traffic shaping as a mechanism to pace TCP throughput.

As stated in Section 5.6.2, the actual throughput achieved by TCP applications depends on two main factors:

- The size of the advertised window determines the transmission rate of the transmitter (disregarding the congestion window).
- The application has to provide data to the socket buffer that the TCP stack can actually fall into the so called steady-state.

There has been some discussion as to whether shaping of TCP traffic, i.e., TCP pacing, might increase fairness and throughput [82, 1]. However, none of these studies was concerned with TCP flows using a GR service. Pacing TCP traffic in an environment offering

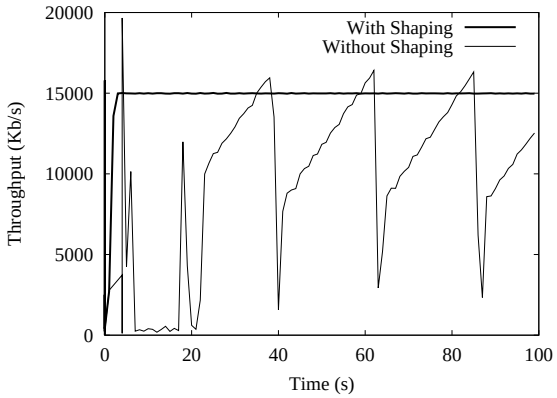

Figure 6.42: Throughput for a GR TCP flow exceeding the reservation. The curves plot the rate reported by the receiver on a per-second base. The SACK-option was disabled. The average throughput achieved was 9440 Kb/s without shaping and 14320 Kb/s when shaping was activated.

a GR service facilitates the simple use of network reservations even without the knowledge of the actual rate the application is writing data to the socket buffer.

So far, the use scenario for a GR service (including the bulk transfer) was based on an instrumented application which was capable of adapting its socket buffer write frequency to the available rate. In a scenario where TCP pacing is controlled by the resource manager of GARA, the use of an oversized socket buffer leverages TCP's self clocking feature to control the speed of transmission. In coordinating a reservation with the shaping rate, packet drops can be avoided and a well-defined throughput can be established.

The following experiment is designed to demonstrate that shaping a TCP flow enables it to work smoothly with a GR service without too much effort. It demonstrates two GR TCP flows between Chicago (ANL) and California (LBNL) which try to exceed the rate they have reserved. This is a fairly likely scenario, since it is often hard for programmers to estimate the bandwidth their applications use. Also, we have already shown elsewhere [115, 57] that applications that do not exceed their rate do not have a problem.

Specifically, the application used a socket buffer size of 1MB. Each flow ran at different times but are shown on the same graph. The application tried to write to the socket buffer at 64 Mb/s while it only made a reservation for 16 Mb/s. One of the flows is shaped to match the reservation bandwidth, i.e. it avoids packet drops. Figure 6.42 shows the achieved throughput for each of the flows. There are two things to notice in this figure. First, the shaped flow has a steady bandwidth at the reservation it made. Second, the unshaped flow has an unstable instantaneous bandwidth. Although it is not obvious from the graph, the average bandwidth of the flow is 9440 Kb/s, which is significantly less than the reservation.

One might hope that using selective acknowledgments would eliminate the need for shaping. This is because SACK can recover from multiple packet losses roughly in one round trip. However, as can be seen from Figure 6.43, this is not the case. In this Figure, we re-

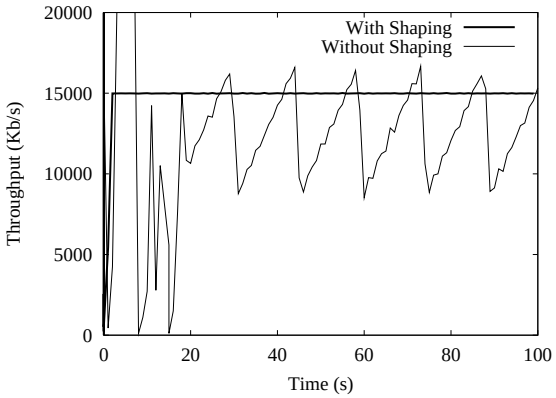

Figure 6.43: Throughput for a GR TCP flow exceeding the reservation. The curves plot the rate reported by the receiver on a per-second base. This time, the SACK-option was enabled. The average achieved throughput was 11992 Kb/s without shaping and 14448 Kb/s when shaping was activated.

peated the same experiment as in Figure 6.42, but with selective acknowledgments enabled. In this case, the instantaneous bandwidth varies much less, but the average bandwidth is still significantly less than the reservation, i.e. 11992 Kb/s compared to 14448 Kb/s. Even though SACK can recover from packet losses more easily, it still interprets the dropped packets as congestion and reacts on this.

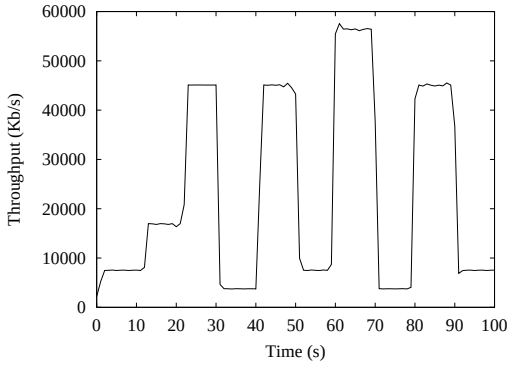

Figure 6.44: Throughput for a GR TCP flow paced by a varying flow-based traffic shaping configuration. The curve plots the rate reported by the receiver on a per-second base. Note that the underlying hardware implements traffic shaping in term of intervals. It therefore allows bursts of one percent of the assigned rate.

The results demonstrate that bulk transfer operation can be controlled by TCP pacing in a GR service. Without detailed knowledge about the rate the application is sending at, the bandwidth broker can provide a constant, smooth transfer rate. Of course, a convenient shaping buffer has to be available in the edge router. This, however, is a realistic assumption as commodity networking devices offer buffer space in the order of megabytes [115].

The results also present the benefit of using SACK. While the experiment without using the SACK-option was oscillating due to packet loss, SACK could reduce its amplitude. However, the achieved throughput with SACK was significantly below the paced experiment.

So far, the presented TCP pacing experiments did not adapt their traffic shaping configuration during the lifetime of the application. Figure 6.44 shows an experiment performed at the testbed in Jülich which demonstrates the ability to dynamically pace a TCP flow by modifying the per-flow based traffic shaping configuration. Here, the application used an oversized socket buffer of 128 kilobytes. Note that even though the application did not perform any adaptation, the bandwidth broker was able to handle significant rate updates by updating the traffic shaping configuration on the ingress router.

6.6.3 TCP Pacing for Distributed Supercomputer Applications

As stated in Section 2.3.1, the traffic profile of a distributed supercomputing application is often quite bursty. We already introduced an example scenario in Figure 2.3. In the illustrated application, messages were transmitted roughly every 60 milliseconds. The typical message size was 128KB with a periodical burst of 256KB. Of course, this pattern is just an example, as the actual traffic profile depends on the application. Also note that all traces reported in this section were gathered on the source host. The illustrated throughput is thus gathered between host and its output network interface card. The actual traffic injected to the network is additionally limited by the local link speed.

In this section we address the occurrence of significant bursts in a policed GR service.

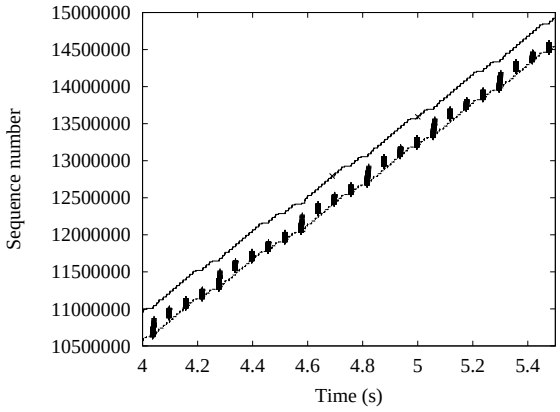

Figure 6.45: TCP sequence numbers over the time within the boundaries of TCP's sliding window. The slope indicates the throughput. The bottom line indicates the behavior of the flow in the aggregate, once traffic shaping was applied. In this scenario, a per-flow based traffic shaping configuration of 34 Mbps was applied.

This inherent burstiness of distributed supercomputer applications is a problem for several reasons. From a resource management point of view, bursts are one of the request parameters which are considered for admission control decisions. Any bandwidth broker will typically limit the amount of bursts injected to the related service aggregate. Of course, a token bucket configuration could be found to police the injected traffic profile appropriately. The depth of the bucket would be related to the maximum message size and the rate

would reflect the average transmission rate. However, the service aggregate may not be designed to support bursts of the requested size, a situation which typically occurs when the EF aggregate is the only available better-than BE service aggregate. Problems also arise from an end-user point of view. Whenever a GR service is used by a distributed supercomputing application, it is unclear what reservation should actually be applied. This is especially important when the access to better-than BE services is associated with additional costs. Access to a GR service is controlled by the specification of two parameters, an average rate and a token bucket depth. In the described scenario, the average transmission rate is not very informative. Due to the fully opened TCP window, each single message will be transmitted link speed, while the average rate might be quite reasonable.

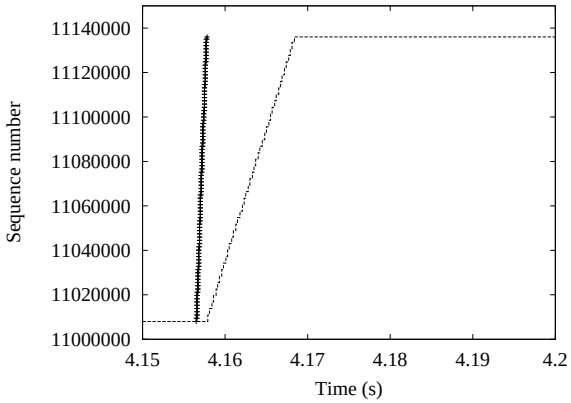

Figure 6.46: TCP sequence numbers and acknowledgments over the time. The slope indicates the throughput. The steep curve lists the outgoing data packets gathered at the network interface card of the source host. The flat curve represents the acknowledgments.

To address this issue, the evolved framework proposed the use of flow-based traffic shaping on the output interface of the source domain's ingress router. In contrast to our configuration for bulk transfers, the underlying idea is not to control and limit the average bandwidth of a flow. Instead, the intention is to support a deterministic and reasonable fast transport of messages. Hence, traffic shaping is used to reduce the burstiness for the traffic entering the aggregate. By adding the ability to actively configure this parameter by means of bursts sizes, the expectation of a guaranteed end-to-end message delay can be build.

Figure 6.45 illustrates the behavior of application 2.3 when traffic shaping is applied. Depending on the actual configuration we recognize a different arrival profile of the TCP acknowledgments (indicated by the bottom line). This arrival profile is an indication for the new data injection profile. We recognize a slope which is less increasing on average when traffic shaping was applied.

Figure 6.46 illustrates the handling of a single burst in a non-congested environment without any traffic shaping. After some delay, the application experiences a constant arrival of acknowledgments. The slope of this arrival rate is caused by the capabilities of the interconnecting network. While the burst was transmitted at link speed, the core network,

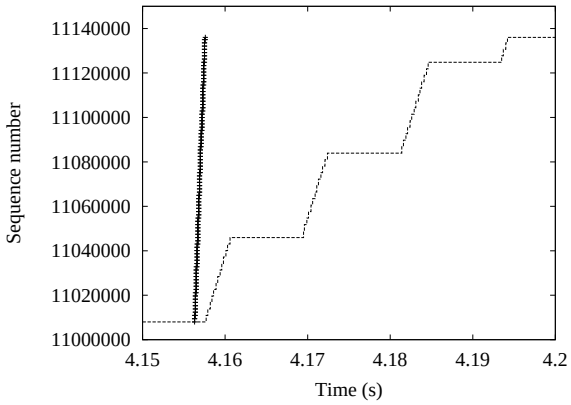

Figure 6.47: TCP sequence numbers and acknowledgments over the time. The slope indicates the throughput. The steep curve lists the outgoing data packets gathered at the network interface card of the source host. The flat curve represents the acknowledgments. Traffic shaping was configured to allow a rate of 32Mb/s.

i.e. the ATM bottleneck link, was not able to handle this speed. We can in fact double-check the configured ATM link capacity by calculating the acknowledgment arrival rate. The acknowledgment of out 128KB bursts in around 18 milliseconds, which relates to the available net capacity of 57 Mb/s.

Figure 6.47 lists the same scenario when traffic shaping was applied. The shaping configuration was allowing a sustained rate of 32Mb/s. This rate was controlled in terms of 10 milliseconds intervals (which is a configurable parameter). The plot illustrates that the burst is split into multiple minor bursts. The reason for this behavior is the implementation of traffic shaping in the Cisco 7200 routers. There, traffic shaping is configured in terms of intervals, mean rates and bursts. During every interval, a maximum of burst size can be sent at any speed. However, the bit rate of the interface will not exceed the mean rate over any integral multiple of the interval. We thus segment the message into bursts of 320 kilobit, or, 40 KB, and guarantee that we handle each segment within the time interval of 10 milliseconds.

Figure 6.48 illustrates the above described scenario when traffic shaping was configured to allow an average rate of 64Mb/s. Due to the division into intervals, this configuration allows the transfer of 640Kb/s in 10 milliseconds while the configuration illustrated in Figure 6.47 limited the bursts to 640 kilobit.

The ability of the application to control traffic shaping facilitates the trade off between message delay and associated service costs. By configuring traffic shaping on-demand, the injected load to the GR service can be adapted to the need. Whenever the GR service is also capable of providing delay boundaries, or is implemented by the EF PHB, application developers can use this information to balance their computation efficiently. A careful admission control which limits the injected traffic is always a prerequisite for this.

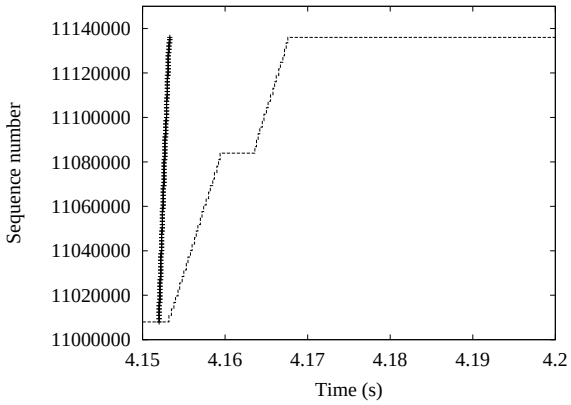

Figure 6.48: TCP sequence numbers and acknowledgments over the time. The slope indicates the throughput. Note that the data output represent the time the source host wrote to the network interface card. Traffic shaping was configured to allow a rate of 64Mb/s.

6.7 Conclusion

This chapter evaluated the proposed bandwidth broker architecture. Using a prototype implementation, it presented a thorough evaluation of the services in different Differentiated Services (DS) environments. All experiments confirmed the capabilities intended with the design, even in an environment which consists of a single better-than best-effort aggregate. The robustness of the provisioned services was impressively improved by the link protection feature of MPLS.

The experiments also demonstrated that the active use of flow-based traffic shaping capabilities actually allows an efficient control of the injected load and is additionally a convenient vehicle to pace TCP-based applications. An efficient use of a Guaranteed Rate service is thus possible, even without instrumentation of the application.

The proposed bulk transfer decision procedure with deadline support was successfully applying for unused bandwidth in both service classes: the Guaranteed Rate class and the best-effort—or more friendly, the less-than best-effort—class. However, the experiments indicated problems with the rate adaptation within the application. This leads to the alternative approach which is based on rate adaptation within the service itself, by incorporating the ability to pace TCP traffic. In addition to its typical use to create a flow that conforms to the reservation, flow-based traffic shaping was also demonstrated in the context of distributed supercomputing applications where it facilitates an effective trade off between the guaranteed message delay and the associated costs.

Chapter 7

Conclusion

This dissertation evolved a flexible bandwidth broker architecture that addresses the unique requirements of emerging Grid applications. The complex trust relationships and usage policies that can apply in Grid environments were solved by an approach, in which network reservations are interpreted as a single reservation step which is controlled by the bandwidth broker of the source domain. In the presented architecture, individual bandwidth brokers communicate via bilaterally authenticated channels between peered domains. Starting with the source domain's bandwidth broker the service request is propagated to the peered downstream entity until it reaches the bandwidth broker of the end-domain. The lack of a transitive trust relation between source- and end-domain is addressed by a delegation model where each bandwidth broker on the path is able to identify all upstream partners by accessing the credentials of the full delegation chain. This procedure allows each bandwidth broker to independently enforce local policies. The accomplishment of this design is the incorporation of assignable capabilities of the Grid resource "network" into an advance reservation framework of the Grid.

All service guarantees were successfully demonstrated in environments which were using standard IP-based QoS building blocks. In this context, the presented architecture extends the functionality of the ingress router of the source domain by additional packet differentiation within a single aggregate. The benefit of this extension is significant. If necessary, it facilitates the support of elastic flows in an expedited forwarding aggregate. By carefully correlating the policing function with the configuration of the shaper, the role of a bandwidth broker is redefined. It is not just a middleware service which provides access to service guarantees in the network, it is also a vehicle to control the behavior of distributed applications. The use of flow based traffic shaping allows the effective control of the traffic injected to a Guaranteed Rate tunnel and is a mechanism to pace TCP flows. Combined with the traffic policing configuration it allows end-users to optimize the requested service parameters with respect to the associated costs and the desired service gain without having to instrument their application.

The proposed integration of the Multiprotocol Label Switching architecture offers additional traffic engineering capabilities which allow the use of external constraint-based routing algorithms to improve the services. First, potential bottleneck links can be avoided by mapping the route of new requests to less heavily used links. As a consequence, the bandwidth broker can serve more service requests. Furthermore, traffic engineering is an additional mechanism for service differentiation. It therefore supports the provision of multiple services on top of a single aggregate. Finally, it is essential for the specification of strong service parameters in a single domain when aggregate scheduling is performed. Applying network calculus, a formal method for the worst-case analysis of the achievable network service, we derived a set of formulas which can be used to calculate the achievable delay-boundary for a path candidate of an external routing algorithm.

Finally, the concept of providing two elementary better-than best-effort services was extended by the support of adaptive techniques. The support of deadline file transfers was incorporated based on a Guaranteed Rate service, offering the minimum capacity required to meet the deadline, and by acquiring additional unused bandwidth of the Guaranteed Rate (GR) service. An effective support of rate adaption was again found by the ability to externally pace TCP flows. Potential scalability problems of the adaptive use of the GR service were addressed by applying the concept to aggregated reservations. Creating a core tunnel between the egress router of the source domain and the ingress router of the destination domain leaves the control about aggregate updates to the bandwidth brokers of the end-domains and therefore avoids rate adaption initiated by state updates in transient domains. It also correlates to the approach to incorporate traffic engineering mechanisms in transient domains, i.e. the allocation of a core tunnel is typically associated with the allocation of an Multiprotocol Label Switching tunnel. While the adaptive use of GR bandwidth assures to meet the deadline, the extension serves file transfers as a hierarchical data flow and improves the economic use of the GR resource. By splitting the relevant files into chunks of data and applying for both, the GR service and the Scavenger service, a less-than best-effort service which is intended to consume unused best-effort capacity, the load to the GR service could significantly be reduced. Highly-tuned file transfer programs improve the gain of the additional use of the Scavenger resource.

Deploying the proposed architecture is an important step to incorporate the network into the Grid resource management. Efficient domain dependent constraint based external routing algorithms must be evolved with the goal, to incorporate these into the control procedures of the bandwidth broker. Once end-to-end deployment is achieved, an important building block for an efficient use of the Grid is accomplished.

List of Acronyms

AF	Assured Forwarding
AIMD	Additive Increase, Multiplicative Decrease
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BE	Best Effort
CAS	Community Authorization Server
CIR	Committed Information Rate
COPS	Common Open Policy Service
CPU	Central Processing Unit
DF	Do not Fragement bit
DN	Distinguished Name
DS	Differentiated Services
DSCP	Differentiated Services Coding Point
EF	Expedited Forwarding
E-LSP	EXP-inferred Label Switched Path
ESnet	Energy Sciences network
EXP	Experimental-Field
FIFO	First In First Out
GARA	General-purpose Architecture for Reservation and Allocation
GARNET	Globus Advance Reservation Network Testbed
GOP	Group of Pictures
GR	Guaranteed Rate
GPS	Generalized Processor Sharing
GRAM	Globus Resource Allocation Manager
HTC	High-Throughput Computing
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force

IGP	Interior Gateway Protocol
IP	Internet Protocol
IS	Integrated Services
ITU	International Telecommunication Union
KB	KiloByte
Kb/s	Kilobit per second
LDAP	Lightweight Directory Access Protocol
LDP	Label Distribution Protocol
L-LSP	Label-inferred Label Switched Path
LRAM	Local Reservation and Allocation Manager
LSP	Label Switched Path
MB	MegaByte
Mb/s	Megabit per second
MPEG4	Moving Picture Experts Group 4
MPI	Message Passing Interface
MPLS	MultiProtocol Label Switching
MRI	Magnetic Resonance Imaging
MTU	Maximum Transmission Unit
NWS	Network Weather Service
OC	Optical Carrier
OSPF	Open Shortest Path First
PDB	Per-Domain Behavior
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PHB	Per-Hop Behavior
PIR	Peak Information Rate
PQ	non-preemptive Priority Queuing
PSRG	Packet Scale Rate Guarantee
PVC	Permanent Virtual Circuit
QoS	Quality of Service
RAA	Resource Allocation Answer
RAR	Resource Allocation Request
RED	Random Early Detection
RSL	Resource Specification Language
RSVP	Resource ReSerVation Protocol

RTT	Round-Trip Time
SACK	Selective ACKnowledgment
SIBBS	Simple Inter-domain Bandwidth Broker Signaling
SLA	Service Level Agreement
SLS	Service Level Specification
SRTCM	Single Rate Three Color Maker
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TE	Traffic Engineering
TLS	Transport Layer Security
TLV	Type-Length-Value
TTL	Time-To-Live
UDP	User Datagram Protocol
URL	Universal Resource Locator
VR	Virtual Reality
WFQ	Weighted Fair Queuing
WRED	Weighted Random Early Detection
WWW	World Wide Web

References

[1] A. Aggarwal, S. Savage, and T. Anderson. Understanding the Performance of TCP Pacing. In *INFOCOM (3)*, pages 1157–1165, 2000.

[2] B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke. Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing. In *IEEE Mass Storage Conference*, 2001.

[3] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. *Internet RFC 2581*, 1997.

[4] K. Almes, S. Kalidindi, and M. Zekauskas. A One-way Delay Metric for IPPM. *Internet RFC 2679*, 1999.

[5] W. Almesberger, J.Y. Le Boudec, and T. Ferrari. Scalable Resource Reservation for the Internet. In *IEEE Conference on Protocols for Multimedia Systems –Multimedia Networking*, Nov 1997.

[6] L. Andersson et al. LDP Specification. *Internet RFC 3036*, 2001.

[7] D. Awduche et al. Applicability Statement for Extensions to RSVP for LSP-Tunnels. *Internet RFC 3210*, 2001.

[8] D. Awduche et al. RSVP-TE: Extensions to RSVP for LSP Tunnels. *Internet RFC 3209*, Dec 2001.

[9] J. Bennett, K. Benson, A. Charny, W. Courtney, and J. Le Boudec. Delay Jitter Bounds and Packet Scale Rate Guarantee for Expedited Forwarding. In *Proceedings of the IEEE INFOCOM 2001, Anchorage, Alaska*, April 2001.

[10] Jon C. R. Bennett and Hui Zhang. WF2q: Worst-Case Fair Weighted Fair Queueing. In *INFOCOM (1)*, pages 120–128, 1996.

[11] S. Blake, D. Black, M. Carlson, M. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. *Internet RFC 2475*, 1998.

[12] J. Bolliger. A Framework for Network-aware Applications, 2000. Dissertation th13636, ETH Zürich.

[13] J.-Y. Le Boudec. Network Calculus made Easy. Technical report epfl-di 96/218, Ecole Polytechnique Federale, Lausanne (EPFL), 1996.

[14] J.-Y. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing System for the Internet*. Springer Verlag - LNCS 2050, 2000.

[15] R. Braden, D. Clark, and S. Shenker. RFC 1633: Integrated Services in the Internet Architecture: an Overview. *Internet RFC 1633*, July 1994.

[16] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP)-version 1 Functional Specification. *Internet RFC 2205*, Sep 1997.

[17] R. Bruyeron, B. Hemon, and L. Zhang. Experimentations with TCP Selective Acknowledgment. *Computer Communication Review*, 28(2):54–77, April 1998.

[18] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch. A National-Scale Authentication Infrastructure. *IEEE Computer*, 33(12):60–66, 2000.

[19] J. Mc Cabe. Network Quality of Service Characterization and Architecture, 1997. `http://www.nas.nasa.gov/Groups/WAN/documents/services-white-paper.html`.

[20] M. Campanella et al. Specification and Implementation Plan for a Premium IP Service, 2001. `http://www.dante.net/geant/GEA-01-032.pdf`.

[21] A. Charny and J.-Y. Le Boudec. Delay Bounds in a Network with Aggregate Scheduling. In *Quality of Future Internet Services, International Workshop, QofIS 2000*, volume 1922, pages 1–13. Springer, 2000.

[22] A. Charny et al. Supplemental Information for the New Definition of the EF PHB (Expedited Forwarding Per-Hop Behavior). *Internet RFC 3247*, 2002.

[23] P.F. Chimento et al. QBone Bandwidth Broker Architecture. Final report available from `http://qbone.internet2.edu/bb/`.

[24] H. Chu and K. Nahrstedt. CPU Service Classes for Multimedia Applications. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*. IEEE Computer Society Press, 1999.

[25] D. Comer. *Internetworking with TCP/IP*. Prentice-Hall International Editions, 1988.

[26] R.L. Cruz. A Calculus for Network Delay, Part ii: Network Analysis. In *IEEE Transactions on Information Theory*, volume 37, pages 132–141, 1991.

[27] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A Resource Management Architecture for Metacomputing Systems.

In *The 4th Workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82. Springer-Verlag LNCS 1459, 1998.

[28] K. Czajkowski, I. Foster, and C. Kesselman. Resource Co-Allocation in Computational Grids. In *Proc. 8th IEEE Symp. on High Performance Distributed Computing*, pages 219–228, 1999.

[29] T. DeFanti and R. Stevens. Teleimmersion. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a Future Computing Infrastructure*, pages 131–155. Morgan Kaufmann Publishers, 1998.

[30] M. Degermark, T. Kohler, S. Pink, and O. Schelen. Advance Reservations for Predictive Service in the Internet. *ACM/Springer Journal of Multimedia Systems*, May 1997.

[31] C. Demichelis and P. Chimento. Ip Packet Delay Variation Metric for IPPM. *Internet Draft draft-ietf-ippm-ipdv-07.txt*, January 2001.

[32] T. Dierks and C. Allen. The TLS Protocol, Version 1.0. *Internet RFC 2246*, January 1999.

[33] S. Dröttboom. Evaluation von Mechanismen zur Unterstützung von Dienstgarantien in Backbone-Netzen. Diploma Thesis, Rheinisch-Westfälische Technische Hochschule Aachen, 2002.

[34] D. Durham et al. The COPS (Common Open Policy Service) Protocol. *Internet RFC 2748*, 2000.

[35] D. G. Dutt, N. Elfassy, D. Durham, and K. McGloghrie. COPS Extensions for RSVP Receiver Proxy. *Internet Draft draft-nitsan-cops-rsvp-proxy-03.txt*, July 2001.

[36] B. Jamoussi (Editor). Constraint-Based LSP Setup using LDP. *Internet RFC 3212*, 2002.

[37] Th. Eickermann, W. Frings, S. Posse, G. Goebbels, and R. Völpel. Distributed Applications in a German Gigabit WAN. In *Proceedings of the eighth IEEE International Symposium on High Performance Distributed Computing*, 1999.

[38] K. Fall and S. Floyd. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. *Computer Communication Review*, 26(3):5–21, July 1996.

[39] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization. *Internet Draft draft-ietf-pkix-ac509prof-09.txt*, June 2001.

[40] D. G. Feitelson. A Survey of Scheduling in Multiprogrammed Parallel Systems. Technical report, IBM Research Report RC19790 (87657), 1995.

[41] P. Ferguson and G. Huston. Quality of Service in the Internet: Fact, Fiction, or Compromise? In *Proceedings of the INET'98*, 1998.

[42] D. Ferrari, A. Gupta, and G. Ventre. Distributed Advance Reservation of Real-Time Connections. *ACM/Springer Verlag Journal on Multimedia Systems*, 5(3), 1997.

[43] D. Ferrari and J. Ramaekers. Client-Network Interactions in Quality of Service Communication Environments. In *Proc. 4th IFIP Conference on High Performance Networking*, 1992.

[44] T. Ferrari and P. Chimento. A Measurement-based Analysis of Expedited Forwarding PHB Mechanisms. In *Proceedings of IWQoS 2000*, pages 127–137, 2000.

[45] V. Firoiu, J.-Y. Le Boudec, D. Towsley, and Z.-L. Zhang. Advances in Internet Quality of Service. Technical report, EPFL, 2001. `dscwww.epfl.ch/EN/publications/documents/tr01_049.pdf`.

[46] F. Fitzek and M. Reisslein. MPEG–4 and H.263 Video Traces for Network Performance Evaluation. *IEEE Network*, 15(6):40–54, November/December 2001.

[47] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A Directory Service for Configuring High-performance Distributed Computations. In *Proc. 6th IEEE Symp. on High Performance Distributing Computing*, pages 365–375. IEEE Computer Society Press, 1997.

[48] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking, V.1 N.4, August 1993*, 1993.

[49] I. Foster, J. Insley, G. von Laszewski, C. Kesselman, and M. Thiebaux. Distance Visualization: Data Exploration on the Grid. *IEEE Computer Magazine*, pages 36–43, December 1999.

[50] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997.

[51] I. Foster and C. Kesselman. The Globus Project: A Status Report. In *Proceedings of the Heterogeneous Computing Workshop*, pages 4–18. IEEE Computer Society Press, 1998.

[52] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann Publishers, 1999.

[53] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A Security Architecture for Computational Grids. In *Proc. 5th ACM Conference on Computer and Communications Security Conference*, pages 83–92, 1998.

[54] I. Foster, C. Kesselman, and S. Tuecke. The Nexus Task-parallel Runtime System. In *Proc. 1st Intl Workshop on Parallel Processing*, pages 457–462. Tata McGraw Hill, 1994.

[55] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *to be published in International Journal Supercomputer Applications*, 2001.

[56] I. Foster, A. Roy, and V. Sander. A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation. In *8th International Workshop on Quality of Service (IWQoS 2000)*, pages 181–188, 2000.

[57] I. Foster, A. Roy, V. Sander, and L. Winkler. End-to-End Quality of Service for High-End Applications. Technical report, Argonne National Laboratory, 1999. `http://www.mcs.anl.gov/qos/qos_papers.htm`.

[58] S. Garfinkel. *PGP: Pretty Good Privacy*. O'Reilly and Assiciates, 1994.

[59] Global Grid Forum. `http://www.gridforum.org`.

[60] The GriPyN Project. `http://www.griphyn.org`.

[61] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard. *Parallel Computing*, 22:789–828, 1996.

[62] R. Guérin and H. Schulzrinne. Network Quality of Service. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a Future Computing Infrastructure*, pages 479–503. Morgan Kaufmann Publishers, 1998.

[63] A. Gurtov. TCP Performance in the Presence of Congestion and Corruption Losses. Master's thesis, Department of Computer Science, University of Helsinky, 2000.

[64] A. Hafid, G. Bochmann, and R. Dssouli. A Quality of Service negotiation Approach with Future Reservations (NAFUR): A Detailed Study. *Computer Networks and ISDN Systems*, 30(8), 1998.

[65] G. Hasegawa, M. Murata, and H. Miyahara. Performance Evaluation of HTTP/TCP on Asymmetric Networks. *International Journal of Communication Systems*, 12(4):281–96, July/August 1999.

[66] J. Heinanen, T. Finland, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. *Internet RFC 2597*, 1999.

[67] J. Heinanen and R. Guerin. A Single Rate Three Color Marker. *Internet RFC 2697*, 1999.

[68] J. Heinanen and R. Guerin. A Two Rate Three Color Marker. *Internet RFC 2698*, 1999.

[69] J. Hoe. Startup Dynamics of TCP's Congestion Control and Avoidance Schemes. Master's Thesis, Massachusetts Institute of Technology, 1995.

[70] J. K. Hollingsworth and S. Maneewongvatana. Imprecise Calendars: an Approach to Scheduling Computational Grids. In *International Conference on Distributed Computing Systems*, pages 352–359, 1999.

[71] G. Hoo, K. Jackson, and W. Johnston. Design of the STARS Network Reservation System. Technical report, LBNL, Orlando Lawrence Berkeley National Laboratory, 2000. `http://www-itg.lbl.gov/QoS/homepage.html`.

[72] G. Hoo, W. Johnston, I. Foster, and A. Roy. QoS as Middleware: Bandwidth Broker System Design. Technical report, LBNL, 1999.

[73] F. Hoßfeld. Teraflops Computing: A Challenge to Parallel Numerics. In *Proc. 4th Intl. ACPC Conference*, 1999.

[74] ITU Telecommunication Standardization Sector (ITU-T). The Directory: Authentication Framework. In *Recommendation X.509*, 1997.

[75] K. Jackson. pyGlobus: A Python Interface to the Globus Toolkit. *Submitted to the Special Issue of Concurrency and Computation: Practice and Experience*, 2001. `http://www.cogkits.org/papers/c545python-cog-cpe.pdf`.

[76] V. Jacobson and M. Karels. Congestion Avoidance and Control. In *Proceedings of the SIGCOMM*, 1988.

[77] V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding PHB. *Internet RFC 2598*, 1999.

[78] K. Kar, M. Kodialam, and T.V. Lakshman. Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications. In *INFOCOM (2)*, pages 884–893, 2000.

[79] J. Knowles and D. Corne. Heuristics for Evolutionary off-line Routing in Telecommunications Networks. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 574–581, 10-12 2000.

[80] J. Kohl and C. Neuman. The Kerberos Network Authentication Service (V5). *Internet RFC 1510*, 1993.

[81] S. Köhler and U. Schäfer. Performance Comparison of Different Class-and-Drop Treatment of Data and Acknowledgements in DiffServ IP Networks. *Proceedings of P&QNet*, pages 245–62, 2000.

[82] J. Kulik, R. Coulter, D. Rockwell, and C. Partridge. Paced TCP for High Delay-Bandwidth Networks. In *Proceedings of the Workshop on Satellite-Based Information Systems (WOSBIS)*, 1999.

[83] K. Lakshman, R. Yavaykar, and R. Finkel. Integrated CPU and Network I/O QoS Management in an Endsystem. In *International Workshop on Quality of Service (IWQOS'97)*, pages 167–178, 1997.

[84] T. Lakshman, U. Madhow, and B. Suter. Window-based Error Recovery and Flow Control with a Slow Acknowledgement Channel: A Study of TCP/IP Performance. In *Proceedings of the IEEE INFOCOM*, 1997.

[85] B. Li and K. Nahrstedt. QualProbes: Middleware QoS Profiling Services for Configuring Adaptive Applications. In *Proceedings of IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware 2000)*, 2000.

[86] M. Litzkow, T. Tannenbaum, J. Basney, and M. Livny. Checkpoint and Migration of UNIX Processes in the Condor Distributed Processing system, 1997.

[87] S. Loken and C. McParland. Snap Computing R&D. `http://www.snap.org`.

[88] A. Mankin et al. Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement: Some Guidelines on Deployment. *Internet RFC 2208*, Sep 1997.

[89] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment Options. *Internet RFC 2028*, 1996.

[90] M. Mathis, J. Semke, and J. Mahdavi. Automatic TCP Buffer Tuning. In *Proceedings of ACM SIGCOMM, volume 28, number 4*, 1998.

[91] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. In *Proceedings of ACM SIGCOMM, volume 27, number 3*, 1997.

[92] P. Messina. Distributed Supercomputing Applications. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a Future Computing Infrastructure*, pages 55–74. Morgan Kaufmann Publishers, 1998.

[93] W. E. Nagel, A. Arnold, M. Weber, H. C. Hoppe, and K. Solchenbach. VAMPIR: Visualization and Analysis of MPI Resources. *Supercomputer*, 12(1):69–80, 1996.

[94] B.C. Neuman. Proxy-Based Authorization and Accounting for Distributed Systems. In *Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 283–291, 1993.

[95] K. Nichols and B. Carpenter. Definition of Differentiated Services Per-Domain Behaviors and Rules for their Specification. *Internet RFC 3086*, 2001.

[96] K. Nichols et al. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. *Internet RFC 2474*, 1998.

[97] K. Nichols, V. Jacobson, and L. Zhang. A Two-Bit Differentiated Services Architecture for the Internet. *Internet RFC 2638*, July 1999.

[98] R. Nitzan and B. Tierney. Experiences with TCP/IP over an ATM OC12 WAN. LBNL Report, Lawrence Berkeley National Laboratory, April 1999.

[99] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Troughput: A simple model and its empirical validation. In *Proceedings of ACM SIGCOMM*, pages 303–314, 1998.

[100] A. K. Parekt and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case. *IEEE/ACM Transactions on Networking*, 1-3:344–357, 1993.

[101] V. Paxson. Measurement in the Internet. PhD thesis, University of California, Berkeley, 1997.

[102] Laura Pearlman, Von Welch, Ian Foster, and Carl Kesselman. A Community Authorization Service for Group Collaboration, 2002. IEEE 3rd International Workshop on Policies for Distributed Systems and Networks.

[103] S. Posse et al. A New Approach to Measure Single-Event Related Brain Activity Using Real-Time fMRI: Feasibility of Sensory, Motor, and Higher Cognitive Tasks. *Human Brain Mapping*, 12:25–41, 2001.

[104] J. Postel and J. Reynolds. Telnet Protocol Specification. *Internet RFC 854*, 1993.

[105] B. Raf and R. Bal. Design Issues for User-level Network Interface Protocols on Myrinet. *IEEE Computer*, 31(11):53–60, 1998.

[106] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. A Resource Allocation Model for QoS Management. In *18th IEEE Real-Time System Symposium*, 1997.

[107] W. Reinhardt. Advance Resource Reservation and its Impact on Reservation Protocols. Technical Report, Rheinisch-Westfälische Technische Hochschule Aachen, 1995.

[108] W. Reinhardt. Advance Reservation of Network Resources for Multimedia Applications. *IWACA94, Lecture Notes in Computer Science*, 868:23–34, 1994.

[109] M. Romberg. The UNICORE Architecture: Seamless Access to Distributed Resources. In *Proceedings of the eighth IEEE International Symposium on High Performance Distributed Computing*, pages 287–293, 1999.

[110] E. Rosen et al. MPLS Label Stack Encoding. *Internet RFC 3032*, 2001.

[111] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. *Internet RFC 3031*, 2001.

[112] A. Roy. End-to-End Quality of Service for High-End Applications, 2001. Dissertation, University of Chicago.

[113] A. Roy, I. Foster, W. Gropp, N. Karonis, V. Sander, and B. Toonen. MPICH-GQ: Quality of Service for Message Passing Programs. In *Proceedings of the IEEE/ACM SC2000 Conference*, November 2000.

[114] S. Sahu, D. Towsley, and K. Kurose. A Quantitative Study of Differentiated Services for the Internet. UMass CMPSCI Technical report, University of Massachusetts, Sept. 1999.

[115] V. Sander, I. Foster, A. Roy, and L. Winkler. A Differentiated Services Implementation for High-Performance TCP Flows. *Elsevier Computer Networks*, 34:915–929, 2000.

[116] O. Schelen and S. Pink. Resource Sharing in Advance Reservation Agents. *Journal of High-Speed Networks*, 7(3-4), 1998. Special Issue on Multimedia Networking.

[117] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service. *Internet RFC 2212*, September 1997.

[118] L. Smarr and C. Catlett. Metacomputing. *Communications of the ACM*, 35(6):44–52, 1992.

[119] W. Smith and D. Gunter. Grid Information Service Schema for Grid Events , 2000. `http://www-didc.lbl.gov/GGF-PerfWG/`.

[120] Q. Snell, M. Clement, D. Jackson, and C. Gregory. The Performance Impact of Advance Reservation Meta-scheduling. In *IPDPS 2000 Workshop, Job Scheduling Strategies for Parallel Processing (JSSPP 2000)*. Springer-Verlag LNCS 1911, 2000.

[121] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. *Internet RFC 2001*, 1997.

[122] W. Stevens. *TCP/IP Illustrated, Vol. 1 The Protocols*. Addison-Wesley, 1997.

[123] B. Teitelbaum. Future Priorities for Internet2 QoS. Technical report, Internet2, 2001. `http://qos.internet2.edu/wg/documents.shtml`.

[124] TF-NGN. Next generation networking, 2000. `http://www.terena.nl/task-forces/tf-ngn`.

[125] M. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, and A. Essiari. Certificate-based Access Control for Widely Distributed Resources. In *Proceedings of the Eighth Usenix Security Symposium*, 1999.

[126] B. Tierney. TCP Tuning Guide for Distributed Application on Wide Area Networks, 2001. `http://www-didc.lbl.gov/tcp-wan.html`.

[127] B. Tierney, W. Johnston, J. Lee, and M. Thompson. A Data Intensive Distributed Computing Architecture for Grid Applications. *Eslevier Journal*, 16(5):473–481, 2000.

[128] P. Trimintzios et al. An Architectural Framework for Providing QoS in IP Differentiated Services Networks. In *7th IFIP/IEEE International Symposium on Integrated Network Management (IM 2001)*, 2001.

[129] S. Tuecke, D. Engert, and M. Thompson. Internet X.509 Public Key Infrastructure Impersonation Certificate Profile, 2001. Global Grid Forum, Working Draft draft-ggf-x509-impersonation-06.txt.

[130] J. Voeckler. Solaris 2.x - Tuning Your TCP/IP Stack and More, 1997. `http://www.sean.de/Solaris/tune.html`.

[131] J. Vollbrecht et al. AAA Authorization Application Examples. *Internet RFC 2905*, August 2000.

[132] X. Wang and H. Schulzrinne. Comparison of Adaptive Internet Multimedia Applications. *Institute of Electronics, Information and Communication Engineers Transactions*, E82-B(6):806–818, June 1999.

[133] The Web100 Project, 2001. `http://www.web100.org/`.

[134] B. Wilkinson and M. Allen. *Parallel Programming - Techniques and Applications Using Networked Workstations and Parallel Computers*. Prentice Hall, 1999.

[135] J. Winett. The Definition of a Socket. *RFC 147*, 1971.

[136] L.C. Wolf and R. Steinmetz. Concepts for Reservation in Advance. *Kluwer Journal on Multimedia Tools and Applications*, 4(3), May 1997.

[137] R. Wolski. Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service. In *Proceedings of the sixth IEEE International Symposium on High Performance Distributed Computing*, Portland, Oregon, 1997. IEEE Press.

[138] J. Wroclawski. Specification of the Controlled-Load Network Element Service. *Internet RFC 2211*, September 1997.

[139] J. Wroclawski. The Use of RSVP with IETF Integrated Services. *Internet RFC 2210*, 1997.

[140] Q. Wu and C. Williamson. Improving Ensemble-TCP Performance on Asymmetric Networks. *Proceedings of MASCOTS*, August 2001.

[141] I. Yeom and A. L. N. Reddy. Modeling TCP Behavior in a Differentiated-Services Network. Technical report, TAMU ECE, 1999.

Lebenslauf

31.5.1965	Geboren in Bottrop
1971-1972	Katholische Grundschule Julius-Spriestersbachstraße Remscheid
1972-1975	Edith-Stein Grundschule Oelde
1975-1975	Gemeinschaftsgrundschule Kuchenheim der Stadt Euskirchen
1975-1984	Städtisches Emil-Fischer Gymnasium Euskirchen - Abitur 1994
1984-1987	Ausbildung zum Mathematisch-technischen Assistenten Forschungszentrum Jülich GmbH - IHK Abschluß 1987
1987-1987	Mathematisch-technischer Assistent Institut für Plasmaphysik Forschungszentrum Jülich GmbH
1987-1996	Mathematisch-technischer Assistent Zentralinstitut für Angewandte Mathematik Forschungszentrum Jülich GmbH
1989-1996	Studium der Informatik FernUniversität Gesamthochschule Hagen - Diplom 1996 mit Auszeichnung
1999-2000	Gastwissenschaftler Mathematics and Computer Science Division Argonne National Laboratory, Illinois, U.S.A.
seit 1996	Wissenschaftlicher Mitarbeiter Zentralinstitut für Angewandte Mathematik Forschungszentrum Jülich GmbH

Already published:

Modern Methods and Algorithms of Quantum Chemistry - Proceedings
Johannes Grotendorst (Editor)
NIC Series Volume 1
Winterschool, 21 - 25 February 2000, Forschungszentrum Jülich
ISBN 3-00-005618-1, February 2000, 562 pages
out of print

Modern Methods and Algorithms of Quantum Chemistry - Poster Presentations
Johannes Grotendorst (Editor)
NIC Series Volume 2
Winterschool, 21 - 25 February 2000, Forschungszentrum Jülich
ISBN 3-00-005746-3, February 2000, 77 pages
out of print

Modern Methods and Algorithms of Quantum Chemistry - Proceedings, Second Edition
Johannes Grotendorst (Editor)
NIC Series Volume 3
Winterschool, 21 - 25 February 2000, Forschungszentrum Jülich
ISBN 3-00-005834-6, December 2000, 638 pages

Nichtlineare Analyse raum-zeitlicher Aspekte der hirnelektrischen Aktivität von Epilepsiepatienten
Jochen Arnold
NIC Series Volume 4
ISBN 3-00-006221-1, September 2000, 120 pages

Elektron-Elektron-Wechselwirkung in Halbleitern: Von hochkorrelierten kohärenten Anfangszuständen zu inkohärentem Transport
Reinhold Lövenich
NIC Series Volume 5
ISBN 3-00-006329-3, August 2000, 145 pages

Erkennung von Nichtlinearitäten und wechselseitigen Abhängigkeiten in Zeitreihen
Andreas Schmitz
NIC Series Volume 6
ISBN 3-00-007871-1, May 2001, 142 pages

Multiparadigm Programming with Object-Oriented Languages - Proceedings
Kei Davis, Yannis Smaragdakis, Jörg Striegnitz (Editors)
NIC Series Volume 7
Workshop MPOOL, 18 May 2001, Budapest
ISBN 3-00-007968-8, June 2001, 160 pages

Europhysics Conference on Computational Physics - Book of Abstracts
Friedel Hossfeld, Kurt Binder (Editors)
NIC Series Volume 8
Conference, 5 - 8 September 2001, Aachen
ISBN 3-00-008236-0, September 2001, 500 pages

NIC Symposium 2001 - Proceedings
Horst Rollnik, Dietrich Wolf (Editors)
NIC Series Volume 9
Symposium, 5 - 6 December 2001, Forschungszentrum Jülich
ISBN 3-00-009055-X

Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms - Lecture Notes
Johannes Grotendorst, Dominik Marx, Alejandro Muramatsu (Editors)
NIC Series Volume 10
Winter School, 25 February - 1 March 2002, Rolduc Conference Centre, Kerkrade, The Netherlands
ISBN 3-00-009057-6, February 2002, 548 pages

Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms- Poster Presentations
Johannes Grotendorst, Dominik Marx, Alejandro Muramatsu (Editors)
NIC Series Volume 11
Winter School, 25 February - 1 March 2002, Rolduc Conference Centre, Kerkrade, The Netherlands

ISBN 3-00-009058-4, February 2002, 85 pages

Strongly Disordered Quantum Spin Systems in Low Dimensions: Numerical Study of Spin Chains, Spin Ladders and Two-Dimensional Systems
Yu-cheng Lin
NIC Series Volume 12
ISBN 3-00-009056-8, May 2002, 131 pages

Multiparadigm Programming with Object-Oriented Languages - Proceedings
Jörg Striegnitz, Kei Davis, Yannis Smaragdakis (Editors)
Workshop MPOOL 2002, 11 June 2002, Malaga
NIC Series Volume 13
ISBN 3-00-009099-1, June 2002, 133 pages

Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms - Audio-Visual Lecture Notes
Johannes Grotendorst, Dominik Marx, Alejandro Muramatsu (Editors)
NIC Series Volume 14
Winter School, 25 February - 1 March 2002, Rolduc Conference Centre, Kerkrade, The Netherlands
ISBN 3-00-010000-8, November 2002, DVD

All volumes are available online at http://www.fz-juelich.de/nic-series/.